TRSTimes

Volume 3. No. 2. - Mar/Apr 1990 - $4.00

Covering Model I, III & 4

LITTLE ORPHAN EIGHTY

In the CLOSE#6 column from issue 2.6. (Nov/Dec 1989) I signed off saying that the next time we would meet would be in a brandnew decade, the last of the twentieth century. What I meant, of course, was that the next issue would be forthcoming in 1990.

Certainly, the Jan/Feb 1990 issue went out on time, but it seems my sign-off was wrong. I have been informed that we are NOT in a new decade yet. We are now (1990) in the last year of the 80's decade. The decade of the 90's begins on January 1, 1991.

Why? Well, let's define the word 'decade'. According to Webster's New World Dictionary decade means '*a period of ten years*'. No problem here, we all knew that, right?

Here comes the rub. When our great (or not so great) ancestors, almost 2000 years ago, decided to mark the beginning of modern time, they gave the initial year the number 1. That means that the very first decade started with year 1 and ended (count 10 now) with year 10. The second decade started with year 11 and ended with year 20. Continuing this trend to the 198th decade, it is plain to see that it started on January 1, 1981 and ends on December 31, 1990, thus beginning the 199th decade (the 90's) on January 1, 1991.

This brings on a related matter. For those of you who were planning to celebrate the end of the twentieth century on December 31, 1999, you'd better wait another year. It doesn't end until December 31, 2000. (Sorry about spoiling your party!)

I realize this doesn't have a thing to do about the TRS-80, but I thought you might like to know.

The Jan/Feb 1990 issue almost did not make it on time. Disaster struck. My hard disk with all the TRSTimes files crashed three days before the issue was scheduled to go to the printers for duplication. No, it was not my TRS-80 hard disk (it always works), it was that *blankety-blank* hard disk on my PC-clone.

After the initial shock of not being able to access anything, I took a deep breath, said some words better not repeated here, and began to 'fix' the problem. This proved to be a complete waste of the better part of two days (and nights). The more I 'fixed', the less it worked.

I finally came to the sad conclusion that the TRSTimes files were REALLY lost, so I reformatted the drive. Luckily most of the files still resided on my TRS-80 hard drive; but they were in TRS-80 format, unedited, and a long way from being ready to send to Ventura Publishing.

Well, sitting and feeling sorry for myself didn't get anything done, so I started the whole process over. The first thing was to install the necessary programs on the PC. Then, on the TRS-80, the files were converted to ASCII as needed and copied to floppies, which were then shoved in the PC where TRSCROSS converted the files to PC format. Each file was individually loaded into Word (im)Perfect and edited as quickly as possible, then saved in a format that Ventura could easily read. Time was running out. I was in a hurry and consequently forgot to run the spell-checker.

Calling my printer, I told him my tale of woe. He assured me that he would do the fastest work possible but, being Christmas, he would not be able to start until the morning of the 27th. Well, that gave me four extra days (*with shopping and family gatherings, make that nights*) to construct and print the masters.

One by one the files were transferred to Ventura, and I spent the next nights blowing up headlines, formatting text, inserting graphics and finalizing the layout.

The morning of the 27th found me at the print shop with the masters just finished. As there would be no time for a proof-copy, we decided to make the entire run at one time. I selected the colored cover paper, and then proceeded to make a mistake when choosing the paper for the inside. It turned out to be entirely too heavy, making it not only difficult to fold neatly, but also costing an additional 20 cents per issue in postage.

What I am trying to say is that the looks of the issue did not turn out the way I had hoped - but I'll give it another whirl with this issue - it is a constant learning process.

STOP THE PRESSES

In the Assembly 101 column I mention that I don't expect to see a TRSDOS 6.4. in the near future. Boy, was I wrong!!!

A couple of days before press time, Roy Soltoff of Misosys calls and announces that he is releasing an upgrade to Model 4 LS-DOS 6.3. It is called LS-DOS 6.3.1. and it supports dates from January 1, 1980 through December 31, 2011, as well as many other fine new features. Best of all, in my opinion, is that Roy plainly states that the alleged copy protection of 6.3. is NOT present on 6.3.1. It is not copy protected - period. (*for further details see the MISOSYS ad on the backpage.*)

FINALLY, we'll have a standard Model 4 DOS. I have already ordered my copy and I urge all Model 4 users to do the same.

And now..... **Welcome to TRSTimes 3.2.**

TRSTimes magazine

Volume 3. No. 2. - Mar/Apr 1990

PUBLISHER EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy Beck
Dr. Allen Jacobs
TECHNICAL ASSISTANCE
Members of:
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 20311 Sherman Way, suite 221. Canoga Park, CA. 91306. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1990 subscription rates (6 issues):
UNITED STATES & CANADA:
$18.00 (U.S. currency)
ALL OTHER COUNTRIES:
$23.00 for surface mail, or
$29.00 for air mail.
(U.S. currency only)
Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used. Also, please make sure that your name and address is written legibly on both hardcopy and disk label.

THE MAIL ROOM

PATCHING THE PATCHES

Thank you very much for TRSTimes and for putting together a collection of PD software.

As a reader of Northern Bytes, I used their OYSTER/BAS on my heavily patched TRSDOS 1.3. to make the system files invisible, but listable. Now I find the PATCH utility rejects the system files (they are in the form SYSxx/SYS, and are unprotected). In LDOS and LS-DOS I can use the FED/CMD to find the load addresses of bytes in executable files, but SU+ in the file utility mode does not have that feature. Do you know of any file utility for TRSDOS 1.3. which has that feature? I will try to RENAME SYS00/SYS to SYS00/CMD and see whether it can then be patched again (on a backup copy, of course). The reason I am so attached to my copy of 1.3. is that in the distant past I found a set of patches for the CONVERT utility that displays the name of the file to be converted and the options (Y/N/Q). Of course, I did not keep track of any of the patches that were applied, but my 1.3. is now very user friendly. Does SYSTEM 1.5. format 80 track drives? Just in case I ever become so flush and bold to install them, a fairly long shot.

Willi E.B. Wald
Hamilton, Ontario.

*It is not mentioned in the TRSDOS 1.3. owner's manual, but you can patch system files by using the notation *NN:D instead of the standard filespec when giving the filename. The patch will then act on the system overlay 'NN' on drive 'D'. System 1.5., as is, supports only 40 track drives (single sided and double sided). It should, however, be possible to patch it for use with 80 track drives. How about it - would anyone like to tackle this?*

Ed.

MORE PATCHING BLUES

I have a problem with a patch listed in vol. 2.4., page 28. I tried the second patch (2 lines) and it works correctly on the date prompt. This was done on a backup TRSDOS 1.3. disk (May 1, 1981). Also the fourth patch (re: error codes) works just fine.

However, the third patch (bypassing the date with <ENTER>) just won't work. No matter how many times I enter it, I still get "STRING NOT FOUND". I tried this one BEFORE the second set.

I wonder if this is similar to "PATCHING BLUES" on page 5 of vol. 2.5.

Also, the CAT program patch from page 16, vol. 2.6. works just fine. So, is it me or the patch?

Leonard A. Falevitch
Phoenix, AZ.

Would you believe me if I told you it was Memorex? Seriously, you should feel much better when I admit it to be the patch. After testing it on the Model III, I made two typos when transferring the information to the final file. The address is incorrect, and "D" was typed as "o". Here is the correct patch allowing you to press <ENTER> to the date prompt.

PATCH *0 (ADD=4EB2,FIND=212D51,CHG=C3FE4E)

My apologies.

Ed.

DEEP SKY OBJECTS

Sign me up for another year. What the heck. I like what I have, though it's changed a little in the last year. I added a 15 meg. hard drive. When that became too small, I switched to a 35 meg. drive. It happens to be faster.

I have worked on a project for a while that involves the TRS-80 community. The Saguaro Astronomy Club released a database of deep sky objects and double stars for the IBM and MAC. With the help of a program by Luis Garcia-Barrio and much scutt work, I converted it to run on Profile 4+. Some changing of the screens, and it could run just as well on Profile 3+. Actually, any database program that can read fixed length, undelimited ASCII text can use the data, all 2 megs worth.

I placed it on Luis' TRSLink BBS (215-848-5728), and I believe Tim Sewell took it from there. I broke it into smaller chunks usable by any single sided double density floppy drive. Those with larger capacity can APPEND the data files together.

TRS-80 users that don't know what "deep sky objects" are won't find this interesting. Those that own telescopes, and want to find more star clusters, nebulae, and galaxies will find this the best database of its kind ever produced. It contains over 10,000 objects and is complete to 15th magnitude. Best of all, it is public domain, and it runs well on a TRS-80. Profile seems especially well suited to tackle it. With indexes, it will find any of the 10,368 objects instantly. If anyone has any questions, they can give me a call at 314-329-3344. Steve Coe, one of the organizers of the database, has given me essentially a free hand in adapting and dis-

tributing the database for the TRS-80 world. Like his, my efforts are presented free. Call me, or write to let me know what you think.

Rev. Peter Besenbruch
29 Williams St.
Ft. Leonard Wood, MO. 65473

Another example of the tireless efforts made by a TRS-80 user for the betterment of the TRS-80 community in general. This should stir plenty of interest.

Ed.

DOTWRITER <-----> FONTASY

I use both DotWriter on a Model 4 and ProSoft's newer implementation, Fontasy, on an IBM compatible. I have a fair number of hours invested in developing partial/special fonts containing symbols, odd letters for logos, and what have you. It would be nice to be able to transfer them back and forth, rather than create them twice.

Also, I have tried to log in to Luis Garcia-Barrio's "8/n/1 #4 bulletin board to explore TRSLink with less than success. I use Lindbergh Systems' Omniterm Plus and everything is displayed on the same line. I suspect that there is a way to do it with the TRSDOS COMM program, but so far it has defeated me.

Lawrence C. Tracy
Morrisville, PA

I know of no program that will directly transfer FONTASY fonts to DotWriter and back. Can anyone help?

If I understand your problem with "8/n/1 #4" correctly, you need to issue a carriage return AND a linefeed at your end after each line. Surely OMNITERM must have an option to issue either just a carriage return, or a carriage return AND a linefeed.

The Model 4 COMM program has this option. The manual describes it as follows:

<CLEAR> <SHIFT> #

This command controls Echoing linefeeds. When enabled, any carriage return your computer receives causes a linefeed character to be transmitted back to the other computer.

This command is useful since there are a large number of terminals and computers that treat a carriage return (ASCII 13) and linefeed (ASCII 10) as seperate functions.

When you are communicating with another TRS-80 computer, you can turn OFF this function by pressing <CLEAR> <SHIFT> # *followed by* <CLEAR> -

<CLEAR> <SHIFT> $ *controls the ability of your computer to accept a linefeed. COMM usually ignores the first linefeed after a carriage return, since most computers send both a carriage return and a linefeed.*

In most cases this is not necessary on TRS-80's where an <ENTER> is treated as both a carriage return and a linefeed.

Ed.

TRSLABEL/CMD

Please include the /SRC files on the TRSTimes on DISK on the assembly language programs published in TRSTimes. I wanted the TRSLABEL/CMD for my Model I, but I didn't have time to peck it in. What I did was to use SUPER UTILITY and replace all occurrences of CD 1B 02 with CD 67 44, and that worked fine, but it would have been a little more elegant to edit the source code and assemble it for the Model I. Also, in TRSTimes 2.5. page 6, you replied to Jim Savage referring to TRSLABEL/BAS, that he didn't need to press <SHIFT> <CLEAR> on the Model III. On the DOSes you say you tested, you are correct; however, on LDOS 5.3. it IS necessary! On the Model I with either LDOS 5.1.5. or 5.3a the <SHIFT> <CLEAR> is also necessary, but then users of these DOSes should know this unique difference.

Have a super 1990!
Art McAninch
Borger, TX.

You are absolutely correct about the LDOS differences. Regarding source codes, again you are right - they should be included on the disk - and will be.

Ed.

AW SHUCKS!

As always, I enjoy receiving your magazine - full of quality information. I always get a kick out of LITTLE ORPHAN EIGHTY. You have quite a sense of humor. I look forward to another year of association with you and your fine magazine.

Carol Lightburne
Kingman, AZ

Though it is always nice getting positive feedback, this felt especially good - thanks Carol.

Ed.

Model I, III & 4 (III)

HAVE YOUR TRS-80 CHECK STUDENT TESTS

By David Berg

STANTEST/BAS is a Basic program written for the Model I to check student tests. It should work on the Model III, and on the Model IV in the III mode. Originally written to ease the burden of checking large numbers of standardized tests, such as the CAT (California Achievement Test) or SRA (Science Research Associates), this program can be used to check any test in which the answers are one-digit- or one-letter-type answers. A little over 15K in length, it was written on TRSDOS 2.7DD, a double-density Model I DOS. For years I used this nifty little DOS without realizing that it was actually a TRSDOS 1.3 (Model III) DOS adapted for the Model I. Though it had some enhancements, the main drawback is that it is a hybrid and compatible with no other Model I DOSes. Commercial word processors will not work with it (except TYPITALL?), nor will it drive a hard disk.

STANTEST/BAS, however, will work with other DOSes if a few minor changes are made. It will be explained later.

The starting menu looks like this:

```
(1) MAKE ANSWER-KEY FILE
(2) ENTER STUDENTS AND ANSWERS, BY ROOM
(3) EXTEND ABOVE FILE, NOT FINISHED
(4) ANSWERS ON FILE -- SCORE TESTS
MAKE ANSWER-KEY FIRST TO ESTABLISH SOURCE FOR LATER
INPUT# OF WORKING VARIABLES.
? -
```

Making the answer-key first enables the program to store the number of test sections, their titles, and number of answers in each, thus allowing the program to automatically dimension arrays. The limits of the program, as written, are 20 test sections, 99 answers each.

Choosing number 2 shows a screen display like this:

```
ENTER ANSWERS <1/2/3/4>, <R> = REPRINT SCREEN, <E> =
ERROR OPTION
EX IS EXTRA SPACE FOR LAST-NUMBER ERROR ROUTINE OPTION
1 NAME OF STUDENT, <EN> <> TO QUIT -
```

After the student name is entered (commas may be used), the screen clears and looks like this example:

```
ENTER ANSWERS <1/2/3/4>, <R> = REPRINT SCREEN,
<E> = ERROR OPTION
EX IS EXTRA SPACE FOR LAST-NUMBER ERROR ROUTINE OPTION
SECTION 1 (title)
N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 . . . EX
? -
```

Standardized tests usually have rows of four rectangles for the answers. Students darken one of the rectangles to answer a question. Entering a number (1/2/3/4) corresponds to the rectangle darkened.

More than four numbers (or less) can be used. Letters or other symbols can be entered instead of numbers, such as A/B/C/D, or any letters except <R> which reprints the screen or <E> which calls an error routine.

"N1" means number 1, and so on, and differentiates the question numbers from the answers. The student's answers can now be entered. As each answer is entered, the prompt jumps to the next number. If a student skips an answer or leaves an answer incomplete, a zero could be entered. If a mistake is made, the prompt will be at the next number. Enter <E> to call an error routine which will enable you to correct any answer on the screen. The screen can also be reprinted from within the error routine. Returning from the error routine puts you in the place you were, or any other number may be chosen. Entering <R> reprints the screen which may be necessary if the screen scrolls. The screen can be reprinted from the beginning or from any other number. "EX" stands for extra, an extra number at the end of each section, in case an error is made on the last number. Instead of the program clearing the screen and jumping to the next section, an <E> can be entered to allow correction of the last numbered answer.

Upon completion of section 1, the program goes on to section 2, and so on, until all answers are entered. The student's name and answers are then stored on disk.

The program saves one student test to disk at a time (or loads one student test at a time when checking). This way the computer's memory is not overloaded, and files can be as long as disk space will allow.

The program then prompts for the name of the next student and the cycle continues until all students and their answers are entered and saved to disk.

You might think that entering all those test answers for all those students would be tedious in itself. The truth is that it goes faster than one might think. Have one hand on the numbers and the other on the enter key, and it goes very fast. You aren't looking back and forth at an answer-key. You don't have to worry about whether the answers are right or wrong, or left-out, or whatever. Just enter what the student has. The computer will check it later. The numeric keypad works well for this. It was an option with the Model I and standard on the Models III and IV, of course.

Choosing number 3 allows you to add more students (and their answers) to a file already on disk. Choosing number 4 has the computer retrieve the answer-key and use it to check student tests. At this point the computer does all the work

(except for feeding the printer). As each student test is retrieved and checked, the program prompts for a printout. Before the first printout, a help screen appears which allows options as to the extent and format of the printout. The program has you answer yes or no to five options.

1. DELETE EMPTY LINE BETWEEN SECTIONS?
2. DELETE "* OR - " PRINTOUT?
3. DELETE ACCURACY-CHECK PRINTOUT?
4. LPRINT ACCURACY-CHECK ON SEPARATE SHEET?
5. DELETE DUAL VIDEO-LPRINTER OUTPUT?

The first part of the printout is not optional and looks similar to this example:

```
SMITH, JOHN                                STUTEST9/SEQ:1
SECTION 1 PHONETIC ANALYSIS ---------- SCORE = 4 / 25
SECTION 2 STRUCTURAL ANALYSIS -------- SCORE = 5 / 11
SECTION 3 READING VOCABULARY --------- SCORE = 4 / 15
SECTION 4 READING COMPREHENSION ---- SCORE = 7 / 20
```

Printout option 1 is explained a little later. Printout option 2 allows a listing of the test questions, and prints an "*" for a correct answer or a "-" for an incorrect one, as shown in the following example.

```
SECTION 1 PHONETIC ANALYSIS
1. * 2. - 3. - 4. * 5. - 6. * 7. * 8. - 9. - 10. * 11. * 12. * 13.- . . .
SECTION 2 STRUCTURAL ANALYSIS
1. * 2. - 3. - 4. * 5. - 6. * 7. * 8. * 9. - 10. - 11. - 12. - ..
```

Printout option 3 allows an accuracy check printout similar to the following:

```
ACCURACY CHECK-STUDENT ANSWER/ MARK / CORRECT ANSWER
SECTION 1 PHONETIC ANALYSIS
1. 1*1 2. 1-2 3. 1-3 4. 2*2 5. 2-1 6. 3*3 7. 3-2 8. 3-2 9. 3-1 10.2-4 . . .
```

For example, 1*1: The first number is the student answer. The last number is the answer-key correct answer. The "*" in the middle indicates a correct answer. The "-" in 1-2 indicates that the student darkened in the first rectangle whereas the second rectangle was the correct choice.

Answering yes to Printout option 1 deletes the empty line between sections (optional listings) to get more on a sheet of paper.

Printout option 4 allows a printout of the accuracy-check option on a separate sheet of paper.

Printout option 5 allows deletion of the "echo to screen" flow from the printer. In TRSDOS 2.7DD these are CMD "Z", "ON" and CMD "Z", "OFF" commands included in the Basic lines. Since TRSDOS 2.7DD and TRSDOS 1.3 are so similar, I would assume that 1.3 would have the same commands, but I'm not sure. Using this program under other DOS's, you could say no to this option or delete it altogether (last part of line 915). Under LDOS 5.1.4 and I assume later versions, to get the same result you would need to enter from DOS the LINK *PR TO *DO (link printer to display) command. The echo-to-screen command is turned on and off in the program because with the CMD "Z" command, the echo works both ways, and you would want things going from the printer to the screen and not vice-versa. The LDOS link command, LINK *PR TO *DO, only goes one way, so it could stay active all the time. I don't know very much about the other fine TRS-80 DOS's such as MULTIDOS, DOSPLUS, and NEWDOS80 except what I've heard and read. They seem impressive, and I'm sure they all have commands to do the same thing. Under TRSDOS 2.3, the Model I single-density DOS, the program should work if you do not use the "Extend file" choice or the echo-to-screen option (the CMD "Z" commands). The echo-to-screen is nice, but not essential.

The program stops the printer at the end of single sheets allowing you to reload. Because much of the information sent to the printer consists of small units with a trailing semi-colon which the printer prints until it runs out of room and then goes to the next line, this seems to fool the line-counter at address 16425, and it increments twice instead of once. The tail-end of line 960 of the program listed below readjusts the line-counter.

```
960 -->:CR=CR+1:IFCR=E+1THENCR=1:
LPRINTCHR$(27);CHR$(10):P=PEEK(16425):POKE16425,P-1
```

My printer is a Radio Shack Lineprinter II. Compared to more recent printers, it is very limited. It has a regular and double-sized mode and that is about it: no underlining, italics, proportional printing, etc. But it's dependable and never breaks down. The CHR$(27); is the "escape" code for it as when you go into double-sized letters: CHR$(27);CHR$(14); or back to regular-size:CHR$(27);CHR$(15). Why CHR$(27); works in line 960, I don't know. It was one of those trial-by-error jobs. (Using the regular CHR$(138) resulted in extra line feeds). Perhaps, for other printers, the CHR$(27);CHR$(10) would need to be changed to get a linefeed.

To prevent syntax error messages, if you are using a DOS other than TRSDOS 2.7DD or TRSDOS 1.3 (I'm assuming that 1.3 has CMD "Z"), then delete (or answer no to) printing option 5 in line 915. Delete from lines 920, 1130, 1180, and 5010 the CMD "Z", "OFF" and/or CMD "Z", "ON" statements. Also delete line 5030.

Line 5 of the program gives you the option to call TRASHMAN, a utility sold by PROSOFT which eliminates or greatly reduces string garbage-collection waits.

STANTEST/BAS uses many strings, not only for the answers but for the numbering of the test questions. Reprinting the screen adds to the number of strings. I found that, in testing the program without TRASHMAN, if the test is not a very short one, that the garbage-collection waits effectively curtail the efficiency of the program. The computer seems to freeze-up and the waits are too numerous. If you are entering and checking a lot of tests, this is not acceptable, to say the least. Using TRASHMAN effectively eliminates this problem. The concern here is that persons wishing to use this program may not have TRASHMAN or a similar utility.

The CLEAR 2000 in line 10 of the program seems a little low to me now. When I write programs now I usually CLEAR at least 5000 bytes. If you do not have TRASHMAN, changing the 2000 to a larger number might help, but it would probably not be a cure. If you have TRASHMAN, line 5 may need to be changed.

```
5 TM=0:PRINT"NEED TO CALL "TRASHMAN" <EN> =NO, 1=YES";:
INPUT TM:IFTM=1DEFUSR=-01346:X=USR(2000):IFX<>0STOP
```

The -01346 number refers to the high-memory marker which will probably be different on your machine. From DOS, typing and entering TM executes TRASHMAN which loads into high memory and indicates on the screen which number to use in the DEFUSR statement.

TRASHMAN is a model 1/3/4/4P utility which can be used with different DOS's, and is a "must-have" type of program. I believe that there is a good chance it can still be obtained. I purchased it in November of 1985 for $39.95. Of course, much TRS-80 software is a lot cheaper than it used to be. PROSOFT also markets the famous ALLWRITE word-processor. Alternatively, the November, 1984, issue of 80 MICRO has an article, "Quit Stalling" by Thomas P. Eggarter (page 86) which shows methods to counteract string garbage-collection delays.

STANTEST/BAS doesn't make any physical marks on the test itself. It can, however, produce an item-by-item printout or an even more detailed accuracy-check printout if the students (or teacher) wish to see the exact details of the test.

Does the computer make mistakes? Does it happen that answers on disk sometime get garbled? In my experience with the program it has never happened, not even once -- even when debugging and testing and when corrections were being made to program lines. The input/output of answer-key and student-answer files has seemed to work with absolute accuracy.

Examples of usefulness: A teacher has six classes of twenty-five students. The students have taken a multiple-choice, true-false test. The teacher could use STANTEST/BAS to check the 150 tests. Or a teacher who uses the commercial tests which come with certain publications could check them with this program.

Or a teacher in a self-contained classroom has had the students take a test. Even for one room of students, it is easier and faster (and more interesting) to have your computer check the test.

Type in the program, then to become familiar with it, my suggestion is that you run a short dummy test on it, making a answer-key, and then entering student answers for it. You could try the <R>eprint screen and <E>rror routine options, and observe the various ways the test results can be printed out.

The program does have a few error traps, but it can be defeated, if one tries to defeat it on purpose. Otherwise, it should work as described. It took quite a bit of figuring out to make the screen work as desired. Both the Model I and III (and IV in the III mode) use a 64 x 16 screen. The Basic program itself is, I believe, pretty well thought out. There are one or two GOTO's which may jump further than they should, to easily understand the program flow. Originally written for 10 test sections, I belatedly thought this might be too little. On these ON X GOTO . . . lines, there is a "fall-through" line which jumps to a GOSUB if there are more than 10 sections. If I had to do it all over again, I would put all these lines together in one place. However, once a program is working and debugged, I don't like to mess with it.

I read somewhere about a decade ago, that a computer is best suited for tasks that are tedious and repetitive. Certainly checking papers and tests fit that category. Before writing this program, I wrote a couple of small test-checking programs that were specific to certain tests (Weekly Reader diagnostic silent reading tests). The results were gratifying. STANTEST/BAS is not specific to any test and can be used to check one-number- or one-letter-answer type tests or papers, whether short or long. If you are someone with stacks of tests piled up needing to be checked, you may want to try this labor-saving tool.

STANTEST/BAS

```
5 TM=0:PRINT"NEED TO CALL "TRASHMAN" <EN> =NO, 1=YES";:
INPUTTM :IFTM=1DEFUSR=-01346:X=USR(2000):IFX<>0STOP
10 CLEAR2000:CLS:DEFINTA-Q,S,U-Z:PSN=320:LN=1:VS=1:
DIMS(20), T$(20)
20 PRINT"STANTEST/BAS:1":PRINT:PRINT"(1) MAKE ANSWER-KEY
FILE": PRINT"(2) ENTER STUDENTS AND ANSWERS, BY ROOM":
PRINT"(3) EXTEND ABOVE FILE, NOT FINISHED":PRINT"(4) ANSWERS
ON FILE – SCORE TESTS"
30 PRINT:PRINT"MAKE ANSWER-KEY FIRST TO ESTABLISH SOURCE
FOR LATER INPUT#":PRINT"OF WORKING VARIABLES.":PRINT:
INPUTOPT: IFOPT=4THEN500
40 PRINT:PRINT"LIMIT = 20 TEST SECTIONS/ 99 ANSWERS EACH":
SV=0:PRINT"HAVE NUM/SECTIONS NUM/ANSWERS/IN/EACH BEEN
SAVED TO DISK?":PRINT"<EN> =NO, 1=YES";:INPUTSV
50 IFSV=1PRINT"TO INPUT# NUM SEC/NUM ANS/SEC TITLES FROM
DISK":PRINT"ENTER NAME OF ANSWER-KEY FILE, LESS EXTENTION
";:INPUTFA$:FA$=FA$+"/SEQ:1":PRINT"OPEN "I",1, ";FA$:
OPEN"I",1,FA$
60 IFSV=1PRINT"INPUT#ING SECTION VARIABLES":INPUT#1,SEC:
FORI=1TOSEC:INPUT#1,S(I):NEXTI:FORI=1TOSEC:INPUT#1,T$(I):
NEXTI:PRINT"CLOSE 1":CLOSE1:GOTO80
70 INPUT"HOW MANY TEST SECTIONS";SEC:IFSEC>20THEN70:
ELSE FORX=1TOSEC:PRINT"HOW MANY ANSWERS, SECTION";X;:
INPUTS(X):LINE INPUT" * * * SECTION TITLE * * * ";T$(X):NEXTX
80 CLS:PRINT"10-SECOND DISPLAY":FORX=1TOSEC:PRINT "SEC";X;
T$(X);S(X);" ";:NEXTX:FORDE=1TO5000:NEXTDE:GOSUB3010
100 CLS:PRINT"ENTER ANSWERS <1/2/3/4>, <R> = REPRINT
SCREEN, <E> = ERROR OPTION":PRINT" EX IS EXTRA SPACE FOR
LAST-NUMBER ERROR ROUTINE OPTION":IFOPT=1PRINT"ENTER
CORRECT ANSWERS.":GOTO120
110 PRINT:STU=STU+1:PRINTSTU;" NAME OF STUDENT, <EN>
<> TO QUIT" ;:LINEINPUTNA$:IFNA$=""THEN440
120 PRINT@PSN-128,STRING$(60," "):PRINT@PSN-128,; 'CLEAR LINE
130 FORX=1TOSEC:PRINT"SECTION";X;T$(X):FORH=1TOS(X)+1:
GOSUB3110 'NUM$
135 JMP=JMP+1:IFJMP=1PRINT" ";
140 PRINTNUM$;:IFH=S(X)+1ORJMP=15THEN150:ELSENEXTH
150 FORANS=0TO14:CNT=CNT+1:VD=VD+1
```

```
160 GOSUB3210 'M MULTIPLIER
170 PA=PSN+M*64+ANS*4:
IFPA>960PA=960
180 PRINT@PA,;:INPUTANS$:
TEMP$=RIGHT$(ANS$,1):IFTEMP$="R"
THEN GOSUB3410:ELSEIFTEMP$="E"
THENP2=PA-ANS*4+64:GOSUB3710
190 IFSUBTHENSUB=0:GOTO270
200 ONXGOTO210,215,220,225,230,235,240,
245,250,255
205 GOSUB4110:GOTO270
210 S1$(CNT)=TEMP$:GOTO270
215 S2$(CNT)=TEMP$:GOTO270
220 S3$(CNT)=TEMP$:GOTO270
225 S4$(CNT)=TEMP$:GOTO270
230 S5$(CNT)=TEMP$:GOTO270
235 S6$(CNT)=TEMP$:GOTO270
240 S7$(CNT)=TEMP$:GOTO270
245 S8$(CNT)=TEMP$:GOTO270
250 S9$(CNT)=TEMP$:GOTO270
255 S0$(CNT)=TEMP$
270 IFNOT(CNT=S(X)+1)NEXTANS
280 JMP=0:PRINT:NEXTH
290 CNT=0:VD=0:VS=1:LN=1:
INPUT"PRESS ENTER FOR NEXT TEST
SECTION";EN$:CLS:PRINTNA$:PRINT@PSN-
128,;:NEXTX
300 IFOPT=1PRINT"ENTER NAME OF
ANSWER-KEY FILE, LESS EXTENTION";:
INPUTFA$:FA$=FA$+"/SEQ:1":PRINT"OPEN
"O",1,";FA$:OPEN"O",1,FA$:F$=FA$:GOTO330
310 PRINT"PRESS ENTER TO PUT ANSWERS
ON DISK-FILE FOR";NA$: INPUTEN$:IFSTU=1
INPUT"WHICH ROOM";RM:GOSUB4010
320 IFSTU=1ANDOPT=2PRINT"OPEN "O",1,
";F$:OPEN"O",1,F$:ELSE IFSTU=1ANOPT=3
PRINT"OPEN "E",1, ";F$:OPEN"E",1,F$
330 PRINT"PRINT#ING TEST ANSWERS TO
";F$:IFOPT=1PRINT#1,SEC:FORO=1TOSEC:
PRINT#1,S(O):NEXT:FORO=1TOSEC:PRINT#
1,CHR$(34);T$(O);CHR$(34):NEXTO:GOTO350
340 PRINT#1,CHR$(34);NA$;CHR$(34)
350 FORX=1TOSEC:FORO=1TOS(X):
ONXGOTO360,365,370,375,380,385,390,
395,400,405
355 GOSUB4210:NEXTO:GOTO420
360 PRINT#1,CHR$(34);S1$(O);CHR$(34):
NEXTO:GOTO420
365 PRINT#1,CHR$(34);S2$(O);CHR$(34):
NEXTO:GOTO420
370 PRINT#1,CHR$(34);S3$(O);CHR$(34):
NEXTO:GOTO420
375 PRINT#1,CHR$(34);S4$(O);CHR$(34):
NEXTO:GOTO420
380 PRINT#1,CHR$(34);S5$(O);CHR$(34):
NEXTO:GOTO420
385 PRINT#1,CHR$(34);S6$(O);CHR$(34):
NEXTO:GOTO420
390 PRINT#1,CHR$(34);S7$(O);CHR$(34):
NEXTO:GOTO420
395 PRINT#1,CHR$(34);S8$(O);CHR$(34):
NEXTO:GOTO420
400 PRINT#1,CHR$(34);S9$(O);CHR$(34):
NEXTO:GOTO420
405 PRINT#1,CHR$(34);S0$(O);CHR$(34):
NEXTO
420 NEXTX:IFOPT=1THEN440
430 INPUT"PRESS ENTER FOR NEXT
STUDENT";EN$:GOTO100
440 PRINTF$;" COMPLETED":PRINT"CLOSE
1":CLOSE1:END
500 PRINT"TO SCORE STUDENT TESTS":
INPUT"NAME OF ANSWER-KEY FILE, LESS
EXTENTION ";FA$:FA$=FA$+"/SEQ:1":
PRINT"OPEN "I",1, ";FA$:OPEN"I",1,FA$
510 PRINT"INPUT#ING PROGRAM VARIABLE
S AND CORRECT TEST ANSWERS . . .":
INPUT#1,SEC:FORI=1TOSEC:INPUT#1,S(I):
NEXTI:FORI=1TOSEC: INPUT# 1,T$(I):NEXTI
520 GOSUB3010
530 FORX=1TOSEC:FORI=1TOS(X):
ONXGOTO540,545,550,555,560,565,570,
575,580,585
535 GOSUB4310:NEXTI:GOTO600
540 INPUT#1,A1$(I):NEXTI:GOTO600
545 INPUT#1,A2$(I):NEXTI:GOTO600
550 INPUT#1,A3$(I):NEXTI:GOTO600
555 INPUT#1,A4$(I):NEXTI:GOTO600
560 INPUT#1,A5$(I):NEXTI:GOTO600
565 INPUT#1,A6$(I):NEXTI:GOTO600
570 INPUT#1,A7$(I):NEXTI:GOTO600
575 INPUT#1,A8$(I):NEXTI:GOTO600
580 INPUT#1,A9$(I):NEXTI:GOTO600
585 INPUT#1,A0$(I):NEXTI
600 NEXTX:PRINTFA$;" COMPLETED.":
PRINT"CLOSE 1":CLOSE1
620 PRINT:INPUT"SCORE THE TESTS OF
WHICH ROOM";RM:GOSUB4010 'FILE$
630 PRINT"OPEN "I",1, ";F$:OPEN"I",1,F$
640 PRINT:STU=STU+1:INPUT#1,NA$:
PRINTSTU;" ";NA$:PRINT"INPUT#ING
STUDENT ANSWERS . . .":PRINT
650 FORX=1TOSEC:FORI=1TOS(X):
ONXGOTO660,665,670,675,680,685,690,
695,700,705
655 GOSUB4410:NEXTI:GOTO720
660 INPUT#1,S1$(I):NEXTI:GOTO720
665 INPUT#1,S2$(I):NEXTI:GOTO720
670 INPUT#1,S3$(I):NEXTI:GOTO720
675 INPUT#1,S4$(I):NEXTI:GOTO720
680 INPUT#1,S5$(I):NEXTI:GOTO720
685 INPUT#1,S6$(I):NEXTI:GOTO720
690 INPUT#1,S7$(I):NEXTI:GOTO720
695 INPUT#1,S8$(I):NEXTI:GOTO720
700 INPUT#1,S9$(I):NEXTI:GOTO720
705 INPUT#1,S0$(I):NEXTI
720 NEXTX:PRINT"TEST ANSWERS OF
";NA$;" INPUT#ED."
750 PRINT"STUDENT ANSWERS NOW BEING
COMPARED TO ANSWER-KEY . . ."
760 FORX=1TOSEC:FORCK=1TOS(X):
ONXGOTO770,780,790,800,810,820,830,
840,850,860
765 GOSUB4510:GOTO880
770 IFS1$(CK)=A1$(CK)THENR1=R1+1:
C1$(CK)="*":ELSEC1$(CK)="-"
775 NEXTCK:R(X)=R1:GOTO880
780 IFS2$(CK)=A2$(CK)THENR2=R2+1:
C2$(CK)="*":ELSEC2$(CK)="-"
785 NEXTCK:R(X)=R2:GOTO880
790 IFS3$(CK)=A3$(CK)THENR3=R3+1:
C3$(CK)="*":ELSEC3$(CK)="-"
795 NEXTCK:R(X)=R3:GOTO880
800 IFS4$(CK)=A4$(CK)THENR4=R4+1:
C4$(CK)="*":ELSEC4$(CK)="-"
805 NEXTCK:R(X)=R4:GOTO880
810 IFS5$(CK)=A5$(CK)THENR5=R5+1:
C5$(CK)="*":ELSEC5$(CK)="-"
815 NEXTCK:R(X)=R5:GOTO880
820 IFS6$(CK)=A6$(CK)THENR6=R6+1:
C6$(CK)="*":ELSEC6$(CK)="-"
825 NEXTCK:R(X)=R6:GOTO880
830 IFS7$(CK)=A7$(CK)THENR7=R7+1:
C7$(CK)="*":ELSEC7$(CK)="-"
835 NEXTCK:R(X)=R7:GOTO880
840 IFS8$(CK)=A8$(CK)THENR8=R8+1:
C8$(CK)="*":ELSEC8$(CK)="-"
845 NEXTCK:R(X)=R8:GOTO880
850 IFS9$(CK)=A9$(CK)THENR9=R9+1:
C9$(CK)="*":ELSEC9$(CK)="-"
855 NEXTCK:R(X)=R9:GOTO880
860 IFS0$(CK)=A0$(CK)THENR0=R0+1:
C0$(CK)="*":ELSEC0$(CK)="-"
865 NEXTCK:R(X)=R0
880 NEXTX
885 'HELP SCREEN FOLLOWS LPRINT FOR-
MAT SAMPLE & OPTIONS
890 IFSTU=1CLS:PRINT"NAME FILE$";
TAB(40)"LPRINT FORMAT EXAMPLE":
PRINTTAB(40)"* =RIGHT, - =WRONG":
FORX=1TO2: PRINT"SECTION";X;"
SCORE=":NEXTX:PRINT"ETC.":PRINT
895 IFSTU=1FORX=1TO2:PRINT"SECTION
";X;" . . .":PRINT" 1. * 2. - 3. - 4. * 5. - 6. * 7. * 8.
* 9. * 10. *":PRINT"11. * 12. * . . .":PRINT:
NEXTX:PRINT"ETC. (5 SEC DELAY)":
FORDE=1TO2500:NEXTDE: CLS
900 IFSTU=1PRINT"ACCURACY CHECK STU
ANS / * - / COR ANS":PRINT: FORX=1TO2:
PRINT"SECTION";X;" . . .":PRINT"1. 2*2 2. 1-2
3. 3-2 4. 1-4 5. 3*3 6. 1*1 7. 4*4 8. 1-2":
PRINT"9. 4*4 10. 3-1 . . .":PRINT:NEXTX
905 IFSTU=1PRINT"ETC.":PRINT:EX=0:
YN$=" <EN>=NO, 1=YES":PRINT "SHOW
LPRINT FORMAT EXAMPLE AGAIN?";YN$;:
INPUTEX:IFEX=1THEN890
910 IFSTU=1O1=0:PRINT:PRINT"OPTION 1:
DELETE EMPTY LINE BETWEEN SECTIONS";
YN$;:INPUTO1:O2=0:PRINT"OPTION 2:
DELETE "* OR -" PRINTOUT";YN$;:INPUTO2:
O3=0:PRINT"OPTION 3: DELETE ACCURACY
-CHECK PRINTOUT";YN$;:INPUTO3
915 IFSTU=1O4=0:PRINT"OPTION 4:
LPRINT ACCURACY-CHECK ON SEPARATE
SHEET";YN$;:INPUTO4:O5=0:
PRINT"OPTION 5: DELETE DUAL VIDEO-
LPRINTER OUTPUT";YN$;:INPUTO5
920 E=10:SP$=" ":PRINT:PRINT"IS
LPRINTER READY?":PRINT"PRESS ENTER
TO LPRINT TEST RESULTS FOR ";NA$;:
INPUTEN$: IFO5=0CMD"Z","ON"
925 LPRINTNA$;TAB(30)F$:LPRINT
CHR$(138)
930 FORX=1TOSEC:L=LEN(T$(X)):
IFX>9L=L+1
940 LPRINT"SECTION";X;T$(X);SP$;
STRING$(20-L,45);SP$;"SCORE =";R(X);
"/";S(X):NEXTX:LPRINTCHR$(138):
IFO2=1ANDO3=1THEN1180
950 FORLOOP=1TO2:IFLOOP=1ANDO2=1
THEN1130
960 FORX=1TOSEC:LPRINT"SECTION";X;
T$(X):FORLP=1TOS(X): LP$=STR$(LP):
NUM$=RIGHT$(LP$,2)+".":CR=CR+1:IFCR=
E+1THENCR=1: LPRINTCHR$(27);
CHR$(10):P=PEEK(16425):POKE16425,P-1
```

```
970 ONXGOTO980,990,1000,1010,1020,1030,
1040,1050,1060,1070
975 GOSUB4710:NEXTLP:GOTO1090
980 LPRINTNUM$;:IFLOOP=1
LPRINTC1$(LP);SP$;:ELSELPRINT
S1$(LP);C1$(LP);A1$(LP);SP$;
985 NEXTLP:GOTO1090
990 LPRINTNUM$;:IFLOOP=1
LPRINTC2$(LP);SP$;:ELSELPRINT
S2$(LP);C2$(LP);A2$(LP);SP$;
995 NEXTLP:GOTO1090
1000 LPRINTNUM$;:IFLOOP=1
LPRINTC3$(LP);SP$;:ELSELPRINT
S3$(LP);C3$(LP);A3$(LP);SP$;
1005 NEXTLP:GOTO1090
1010 LPRINTNUM$;:IFLOOP=1
LPRINTC4$(LP);SP$;:ELSELPRINT
S4$(LP);C4$(LP);A4$(LP);SP$;
1015 NEXTLP:GOTO1090
1020 LPRINTNUM$;:IFLOOP=1
LPRINTC5$(LP);SP$;:ELSELPRINT
S5$(LP);C5$(LP);A5$(LP);SP$;
1025 NEXTLP:GOTO1090
1030 LPRINTNUM$;:IFLOOP=1
LPRINTC6$(LP);SP$;:ELSELPRINT
S6$(LP);C6$(LP);A6$(LP);SP$;
1035 NEXTLP:GOTO1090
1040 LPRINTNUM$;:IFLOOP=1
LPRINTC7$(LP);SP$;:ELSELPRINT
S7$(LP);C7$(LP);A7$(LP);SP$;
1045 NEXTLP:GOTO1090
1050 LPRINTNUM$;:IFLOOP=1
LPRINTC8$(LP);SP$;:ELSELPRINT
S8$(LP);C8$(LP);A8$(LP);SP$;
1055 NEXTLP:GOTO1090
1060 LPRINTNUM$;:IFLOOP=1
LPRINTC9$(LP);SP$;:ELSELPRINT
S9$(LP);C9$(LP);A9$(LP);SP$;
1065 NEXTLP:GOTO1090
1070 LPRINTNUM$;:IFLOOP=1
LPRINTC0$(LP);SP$;:ELSELPRINT
S0$(LP);C0$(LP);A0$(LP);SP$;
1075 NEXTLP
1090 CR=0:IFO1=1NN=1:ELSENN=2
1100 FORLF=1TONN:LPRINT" ":NEXTLF:
GOSUB5110
1110 NEXTX
1120 IFLOOP=1ANDO3=1THEN1180
1130 IFLOOP=1ANDO4=1CMD"Z","OFF":
PRINT"RELOAD LPRINTER FOR ACCURACY-
CHECK PRINTOUT. <EN> TO CONT";:
INPUTEN$:POKE16425,1:
IFO5=0CMD"Z","ON"
1140 IFLOOP=1ANDO4=1THEN1160:
ELSEIFLOOP=1ANDO2=1THEN1160
1150 IFLOOP=1FORCR=1TO3:
LPRINTCHR$(138):NEXTCR:CR=0
1160 IFLOOP=1LPRINT"ACCURACY-CHECK
STUDENT ANSWER/ RIGHT OR WRONG
MARK/ CORRECT ANSWER":
LPRINTCHR$(138):E=8
1170 NEXTLOOP
1180 CMD"Z","OFF":POKE16425,1:CPY=0:
PRINT"NEED ANOTHER COPY/ SAME STU-
DENT,";YN$;:INPUTCPY:IFCPY>0THEN920
1190 R1=0:R2=0:R3=0:R4=0:R5=0:R6=0:
R7=0:R8=0:R9=0:R0=0:RA=0:RB=0:
RC=0:RD=0:RF=0:RG=0:RH=0:RI=0:RJ=0
:RK=0: IFNOT(EOF(1))THEN640
1200 PRINT"ALL STUDENTS COMPLETED ";
F$:PRINT"CLOSE 1":CLOSE1:
STU=0:GOTO620
3000 END 'DIM ARRAYS
3010 S1=S(1):S2=S(2):S3=S(3):S4=S(4):
S5=S(5):S6=S(6):S7=S(7):S8=S(8):
S9=S(9):S0=S(10):IFSEC>10SA=S(11):
SB=S(12):SC=S(13):SD=S(14):SF=S(15):
SG=S(16):SH=S(17):SI=S(18):SJ=S(19):
SK=S(20)
3020 DIMS1$(S1+1),S2$(S2+1),S3$(S3+1),
S4$(S4+1),S5$(S5+1),S6$(S6+1),
S7$(S7+1),S8$(S8+1),S9$(S9+1),
S0$(S0+1):IFSEC>10THENDIMSA$(SA+1),
SB$(SB+1),SC$(SC+1),SD$(SD+1),
SF$(SF+1),SG$(SG+1),
SH$(SH+1),SI$(SI+1),SJ$(SJ+1),SK$(SK+1)
3030 IFOPT=4THENDIMA1$(S1),A2$(S2),
A3$(S3),A4$(S4),A5$(S5),A6$(S6),A7$(S7),
A8$(S8),A9$(S9),A0$(S0),C1$(S1),C2$(S2),
C3$(S3),C4$(S4),C5$(S5),C6$(S6),C7$(S7),
C8$(S8),C9$(S9),C0$(S0),R(20)
3040 IFOPT=4ANDSEC>10THEN
DIMAA$(SA),AB$(SB),AC$(SC),AD$(SD),
AF$(SF),AG$(SG),AH$(SH),AI$(SI),AJ$(SJ),
AK$(SK),CA$(SA),CB$(SB),CC$(SC),CD$(SD),
CF$(SF),CG$(SG),CH$(SH),CI$(SI),CJ$(SJ),
CK$(SK)
3050 RETURN
3100 END 'NUM$
3110 IFH=S(X)+1NUM$="EX":RETURN
3120 IFH<10NUM$="N"+RIGHT$
(STR$(H),1)+" ":ELSENUM$="N"
+RIGHT$(STR$(H),2)+" "
3130 RETURN
3200 END 'M MULTIPLIER
3210 IFVD=<15M=0:RETURN
3220 IFVD>15ANDVD=<30M=3:RETURN
3230 IFVD>30ANDVD=<45M=6:RETURN
3240 IFVD>45ANDVD=<60M=9:RETURN
3250 IFVD>60ANDVD=<75M=12:RETURN
3260 IFVD>75ANDVD=<90M=15:RETURN
3300 END 'ADJUST M IF SCREEN
REPRINTED NOT FROM LINE 1
3310 IFNOT(LN=1)M=M-((LN-1)*3)
3320 RETURN
3400 END 'REPRINT SCREEN
3410 PRINT"REPRINT SCREEN – ENTER
NUMBER AT WHICH TO BEGIN.";:
INPUTRE:IFRE<1ORRE>CNTTHEN3410
3420 ADD=RE-1:WH=RE:GOSUB3910
3430 LN=TEMP+1:CIR=1:VD=SHFT:
J2=SHFT:CLS:PRINT@PSN-128,"SECTI
ON";X;T$(X):PRINT@PSN-64+SHFT*4,;
3440 FORH2=RE TO S(X)+1:H=H2:
GOSUB3110 'NUM$
3450 J2=J2+1:IFJ2=1PRINT" ";
3460 PRINTNUM$;:IFH2=S(X)+1OR
J2=15THEN3470:ELSENEXTH2
3470 IFCIR=1BE=SHFT:ELSEBE=0
3480 FORA2=BE TO J2-1:VD=VD+1
:GOSUB3210 'M MULTIPLIER
3490 PA=PSN+M*64+A2*4+1:
IFPA>960PA=960
3500 IFCIR=1A=RE:ELSEA=A+1
3510 CIR=0:ONXGOTO3520,3525,3530,
3535,3540,3545,3550,3555,3560,3565
3515 GOSUB4910:GOTO3580
3520 PRINT@PA,S1$(A);:GOTO3580
3525 PRINT@PA,S2$(A);:GOTO3580
3530 PRINT@PA,S3$(A);:GOTO3580
3535 PRINT@PA,S4$(A);:GOTO3580
3540 PRINT@PA,S5$(A);:GOTO3580
3545 PRINT@PA,S6$(A);:GOTO3580
3550 PRINT@PA,S7$(A);:GOTO3580
3555 PRINT@PA,S8$(A);:GOTO3580
3560 PRINT@PA,S9$(A);:GOTO3580
3565 PRINT@PA,S0$(A);
3580 ADD=ADD+1:IFADD=CNT
THEN3590:ELSENEXTA2:J2=0:PRINT:
PRINT: NEXTH2
3590 SUB=-1:ANS=ANS-1:CNT=CNT-1:
VD=VD-1:P2=PA-A2*4+64-1:VS=RE
3600 RETURN
3700 END 'ERROR ROUTINE
3710 PRINT@P2,"TEST ANSWER TO BE COR
RECTED MUST APPEAR ON SCREEN."
3720 WH=0:PRINT"CHANGE WHICH
ANSWER? <NUM>, <>=EXIT SUB, <-1>
=REPRINT SCREEN";:INPUTWH:
IFWH<0GOSUB3410:ELSEIFWH=0THEN3830

3730 IFSUBTHENSUB=0:GOTO3710
3740 VD=WH:K=WH:GOSUB3210:
GOSUB3310:GOSUB3910
3750 PA=PSN+M*64+SHFT*4:
PRINT@PA,;:INPUTANS$:
TEMP$=RIGHT$(ANS$,1)
3760 ONXGOTO3770,3775,3780,3785,
3790,3795,3800,3805,3810,3815
3765 GOSUB5010:GOTO3710
3770 S1$(WH)=TEMP$:GOTO3710
3775 S2$(WH)=TEMP$:GOTO3710
3780 S3$(WH)=TEMP$:GOTO3710
3785 S4$(WH)=TEMP$:GOTO3710
3790 S5$(WH)=TEMP$:GOTO3710
3795 S6$(WH)=TEMP$:GOTO3710
3800 S7$(WH)=TEMP$:GOTO3710
3805 S8$(WH)=TEMP$:GOTO3710
3810 S9$(WH)=TEMP$:GOTO3710
3815 S0$(WH)=TEMP$:GOTO3710
3830 PRINT"(1) CONTINUE AT PLACE OF
ERROR + 1":PRINT"(2) CONTINUE AT LAST
REGULAR INPUT":PRINT"(3) CONTINUE AT
OTHER NUMBER":INPUTCH
3840 IFCH=1CNT=K+1:ANS=SH
FT+1:RE=VS:GOSUB3420:RETURN
3850 IFCH=2RE=VS:GOSUB3420:RETURN
3860 IFCH=3INPUT"WHICH NUMBER";
WH:CNT=WH:GOSUB3910
3870 ANS=SHFT:RE=VS:GOSUB3420:
RETURN
3900 END 'SHFT VALUE
3910 TEMP=INT((WH-1)/15):
SHFT=(WH-1)-TEMP*15:RETURN
4000 END 'FILE$
4010 PRINT"ENTER NAME OF FILE, LESS
ROOM NUMBER AND EXTENTION (WILL BE
ADDED BY PROGRAM)":INPUTNF$:
RM$=STR$(RM):IFLEN(RM$)=2THENR$=
RIGHT$(RM$,1):ELSER$=RIGHT$(RM$,2)
4020 F$=NF$+R$+"/SEQ:1":RETURN
4100 END '>10 sec, input ans
4110 ONX-10GOTO4120,4125,4130,4135,
4140,4145,4150,4155,4160,4165
4120 SA$(CNT)=TEMP$:RETURN
4125 SB$(CNT)=TEMP$:RETURN
4130 SC$(CNT)=TEMP$:RETURN
```

```
4140 SF$(CNT) = TEMP$:RETURN
4145 SG$(CNT) = TEMP$:RETURN
4150 SH$(CNT) = TEMP$:RETURN
4155 SI$(CNT) = TEMP$:RETURN
4160 SJ$(CNT) = TEMP$:RETURN
4165 SK$(CNT) = TEMP$:RETURN
4200 END '> 10 SEC, PRINT#1, ANS
4210 ONX-10GOTO4220,4225,4230,4235,
4240,4245,4250,4255,4260,4265
4220 PRINT#1,CHR$(34);SA$(O);CHR$(34):
RETURN
4225 PRINT#1,CHR$(34);SB$(O);CHR$(34):
RETURN
4230 PRINT#1,CHR$(34);SC$(O);CHR$(34):
RETURN
4235 PRINT#1,CHR$(34);SD$(O);CHR$(34):
RETURN
4240 PRINT#1,CHR$(34);SF$(O);CHR$(34):
RETURN
4245 PRINT#1,CHR$(34);SG$(O);CHR$(34):
RETURN
4250 PRINT#1,CHR$(34);SH$(O);CHR$(34):
RETURN
4255 PRINT#1,CHR$(34);SI$(O);CHR$(34):
RETURN
4260 PRINT#1,CHR$(34);SJ$(O);CHR$(34):
RETURN
4265 PRINT#1,CHR$(34);SK$(O);CHR$(34):
RETURN
4300 END '> 10 SEC, INPUT#1 ANS/KEY
4310 ONX-10GOTO4320,4325,4330,4335,4340,
4345,4350,4355,4360, 4365
4320 INPUT#1,AA$(I):RETURN
4325 INPUT#1,AB$(I):RETURN
4330 INPUT#1,AC$(I):RETURN
4335 INPUT#1,AD$(I):RETURN
4340 INPUT#1,AF$(I):RETURN
4345 INPUT#1,AG$(I):RETURN
4350 INPUT#1,AH$(I):RETURN
4355 INPUT#1,AI$(I):RETURN
4360 INPUT#1,AJ$(I):RETURN
4365 INPUT#1,AK$(I):RETURN
4400 END '> 10 SEC, INPUT#1 STU/ANS
4410 ONX-10GOTO4420,4425,4430,4435,4440,
4445,4450,4455,4460, 4465
4420 INPUT#1,SA$(I):RETURN
4425 INPUT#1,SB$(I):RETURN
4430 INPUT#1,SC$(I):RETURN
4435 INPUT#1,SD$(I):RETURN
4440 INPUT#1,SF$(I):RETURN
4445 INPUT#1,SG$(I):RETURN
4450 INPUT#1,SH$(I):RETURN
4455 INPUT#1,SI$(I):RETURN
4460 INPUT#1,SJ$(I):RETURN
4465 INPUT#1,SK$(I):RETURN
4500 END '> 10 SEC, CHECK STU/ANS WITH
ANS/KEY
4510 FORCK = 1TOS(X):ONX-10GOTO4520,
4530,4540,4550,4560,4570,4580,4590,4600,
4610
4520 IFSA$(CK) = AA$(CK)THENRA = RA + 1:C
A$(CK) = "*":ELSECA$(CK) = "-"
4525 NEXTCK:R(X) = RA:RETURN
4530 IFSB$(CK) = AB$(CK)THEN
RB = RB + 1:CB$(CK) = "*":ELSECB$(CK) = "-"
4535 NEXTCK:R(X) = RB:RETURN
4540 IFSC$(CK) = AC$(CK)THENRC =
RC + 1:CC$(CK) = "*":ELSECC$(CK) = "-"
4545 NEXTCK:R(X) = RC:RETURN
4550 IFSD$(CK) = AD$(CK)THENRD = RD + 1:
CD$(CK) = "*":ELSECD$(CK) = "-"
4555 NEXTCK:R(X) = RD:RETURN
4560 IFSF$(CK) = AF$(CK)THENRF = RF + 1:
CF$(CK) = "*":ELSECF$(CK) = "-"
4565 NEXTCK:R(X) = RF:RETURN
4570 IFSG$(CK) = AG$(CK)THENRG = RG + 1:
CG$(CK) = "*":ELSECG$(CK) = "-"
4575 NEXTCK:R(X) = RG:RETURN
4580 IFSH$(CK) = AH$(CK)THENRH = RH + 1:
CH$(CK) = "*":ELSECH$(CK) = "-"
4585 NEXTCK:R(X) = RH:RETURN
4590 IFSI$(CK) = AI$(CK)THENRI = RI + 1:
CI$(CK) = "*":ELSECI$(CK) = "-"
4595 NEXTCK:R(X) = RI:RETURN
4600 IFSJ$(CK) = AJ$(CK)THENRJ = RJ + 1:
CJ$(CK) = "*":ELSECJ$(CK) = "-"
4605 NEXTCK:R(X) = RJ:RETURN
4610 IFSK$(CK) = AK$(CK)THENRK = RK + 1:
CK$(CK) = "*":ELSECK$(CK) = "-"
4615 NEXTCK:R(X) = RK:RETURN
4700 END '> 10 SEC, LPRINT
4710 ONX-10GOTO4720,4730,4740,4750,4760,
4770,4780,4790,4800, 4810
4720 LPRINTNUM$;:IFLOOP = 1LPRINT
CA$(LP);SP$;:RETURN:ELSELPRINTSA$(LP);
CA$(LP);AA$(LP);SP$;:RETURN
4730 LPRINTNUM$;:IFLOOP = 1LPRINT
CB$(LP);SP$;:RETURN:ELSELPRINTSB$(LP);
CB$(LP);AB$(LP);SP$;:RETURN
4740 LPRINTNUM$;:IFLOOP = 1LPRINT
CC$(LP);SP$;:RETURN:ELSELPRINT
SC$(LP);CC$(LP);AC$(LP);SP$;:RETURN
4750 LPRINTNUM$;:IFLOOP = 1LPRINT
CD$(LP);SP$;:RETURN:ELSELPRINT
SD$(LP);CD$(LP);AD$(LP);SP$;:RETURN
4760 LPRINTNUM$;:IFLOOP = 1LPRINT
CF$(LP);SP$;:RETURN:ELSELPRINT
SF$(LP);CF$(LP);AF$(LP);SP$;:RETURN
4770 LPRINTNUM$;:IFLOOP = 1LPRINT
CG$(LP);SP$;:RETURN:ELSELPRINT
SG$(LP);CG$(LP);AG$(LP);SP$;:RETURN
4780 LPRINTNUM$;:IFLOOP = 1LPRINT
CH$(LP);SP$;:RETURN:ELSELPRINT
SH$(LP);CH$(LP);AH$(LP);SP$;:RETURN
4790 LPRINTNUM$;:IFLOOP = 1LPRINT
CI$(LP);SP$;:RETURN:ELSE
LPRINTSI$(LP);CI$(LP);AI$(LP);SP$;:RETURN
4800 LPRINTNUM$;:IFLOOP = 1LPRINT
CJ$(LP);SP$;:RETURN:ELSELPRINT
SJ$(LP);CJ$(LP);AJ$(LP);SP$;:RETURN
4810 LPRINTNUM$;:IFLOOP = 1LPRINT
CK$(LP);SP$;:RETURN:ELSELPRINT
SK$(LP);CK$(LP);AK$(LP);SP$;:RETURN
4900 END '> 10 SEC, REPRINT SCREEN
4910 ONX-10GOTO4920,4925,4930,
4935,4940,4945,4950,4955,4960,4965
4920 PRINT@PA,SA$(A);:RETURN
4925 PRINT@PA,SB$(A);:RETURN
4930 PRINT@PA,SC$(A);:RETURN
4935 PRINT@PA,SD$(A);:RETURN
4940 PRINT@PA,SF$(A);:RETURN
4945 PRINT@PA,SG$(A);:RETURN
4950 PRINT@PA,SH$(A);:RETURN
4955 PRINT@PA,SI$(A);:RETURN
4960 PRINT@PA,SJ$(A);:RETURN
4965 PRINT@PA,SK$(A);:RETURN
5000 END '> 10 SEC, ERROR SUB
5010 ONX-10GOTO5020,5025,5030,5035,
5040,5045,5050,5055,5060,5065
5020 SA$(WH) = TEMP$:RETURN
5025 SB$(WH) = TEMP$:RETURN
5030 SC$(WH) = TEMP$:RETURN
5035 SD$(WH) = TEMP$:RETURN
5040 SF$(WH) = TEMP$:RETURN
5045 SG$(WH) = TEMP$:RETURN
5050 SH$(WH) = TEMP$:RETURN
5055 SI$(WH) = TEMP$:RETURN
5060 SJ$(WH) = TEMP$:RETURN
5065 SK$(WH) = TEMP$:RETURN
5100 END 'RELOAD LPRINTER?
5110 P = PEEK(16425):RL = 0:CMD"Z","OFF"
5120 IFP = > 50PRINT"LPRINTER LINE
COUNT = ";P;". RELOAD LPRINTER?":
PRINTYN$;" (ENTER 1 BEFORE RELOAD-
ING)";:INPUTRL:IFRL = 1LPRINTCHR$(11):
INPUT"PRESS ENTER TO CONTINUE";EN$
5130 IFO5 = 0CMD"Z","ON"
5140 RETURN
```

GIF4MOD4

TAKING A SECOND LOOK... AT VERSION 2 (Menu Driven Cinemascope)

By Dr. Allen Jacobs

In TRSTImes 2.6, I revlewed a sophlsticated, valuable, and unique graphlcs program for the TRS-80 Model 4 entitled GIF4MOD4, by G.F.R "Frank" Slinkman. In this issue, I have the pleasure of reviewlng a totally new GIF graphic file conversion utility entitled GIF4MOD4 Version 2. Why do I say it is totally new? Because virtually every aspect of Version 2 has been improved or is a new feature, compared to Version 1. Why compare a program only to an earlier version of itself? Because, previous to Version 2, the only utility of its kind for the TRS-80 was Version 1.

Some information about what this program does, and what GIF files are, can be found in the references at the end of this article. In case you do not have access to these references, I will try to explain the program's overall function.

GIF4MOD4 converts GIF graphics files into displayable images, on either the Radio Shack or Microlabs hi-rez graphics boards. "GIF" stands for the "Graphics Interchange Format" developed by Compuserve for storing hi-rez graphics in a form that enables computer systems normally incompatible with each other to exchange graphics images. GIF graphics files are available from a number of sources in addition to Compuserve itself. Almost every other computer system in the world can create, edit, transfer, and store graphics image files in GIF. Many bulletin board systems and public domain software houses store libraries of GIF graphics image files.

Within my expectations, rudimentary GIF file image recovery should have been enough of a task for any self respecting TRS-80 utility. This is because, from the point at which a hi-rez graphics image is produced, it can be edited by any TRS-80 hi-rez paint/draw program. It can be stored in any format supported by the editing program (/HR, /HRG, /CHR, /BLK, etc. formats), and it can also printed. PRO-DRAW, REMBRANT, and TRS-DRAW are the names of programs that immediately come to mind for performing these functions.

The problems of shading and color

If shading is a rather easy task for a black-and-white television receiving standard color composite video broadcast signals, it would seem that there should be no problem for a program processing a GIF file to one of our monochrome hi-rez boards to achieve a perfect monochrome rendering. Such is not the case.

All television broadcasts are designed to a single minimum standard that was adopted before the first commercial television receiver was ever made. Therefore, a black-and-white television receiving standard color composite video broadcast signals requires no image reconstruction processing because the format of the signals it receives are transmitted with the luminance information for the image separate from the chrominance information. The luminance information is the brightness of each area in the image while the chrominance information is the actual color for that area.

All the monochrome television set has to do is to ignore the color information as it processes the video signal and you have a perfectly respectable "black-and-white" (monochrome) image. In the case of composite video (normal television), color is "add-on" information to an originally monochrome signal.

This broadcast standard was established, by the FCC, with the requirement that future enhancements to the original monochrome standard would not require additional processing or equipment by existing, older receivers. For this reason, a standard for high resolution television broadcasts has not, as yet, been adopted. The bad news is that it may be years until such a standard is agreed upon. The good news is that when it is finally established, all receivers will be able to receive the same quality of images that they received before adoption of the new standard.

In the case of computer graphics, numerous display systems, incompatible with each other, were in existence before any graphics display standard was established. None of these systems are under any regulatory requirement to comply to any standard display format. In this case, an interchange format, such as GIF, has to accommodate for all the existing incompatibilities and provide for others that have not, as yet, been developed. It has to do all this and still have a manageable file size. This after the fact format has only one factor working in its favor. It is a computer format. The processing necessary to tailor a file adhering to the format can be performed by the receiving computer itself, without the addition of any hardware, as long as there is software capable of accomplishing the task.

Since the GIF file format was developed in 1987, long after the advent of color and digitally encoded imaging techniques, it has become more efficient to encode only the color information for each discrete area of a graphics image. The name for a discrete area in a digitally encoded image is called a pixel. The GIF format allows the computer to select a color for each pixel, which automatically assigns the inherent brightness of the selected color, to that pixel.

While the current version of GIF (V87a) format allows images to contain as many as 256 different colors, what do you do if you have only two "colors" available to you (black and white) and no way of varying the brightness of a pixel on the screen? You are able to produce only high contrast images or line drawings. This is the case with our hi-rez monochrome boards. While this level of control is acceptable for some cartoons, it is not adequate for working with pictures originally created in color.

What makes GIF4MOD4 so sophisticated is that it automatically shades the monochrome image it produces on our hi-rez boards with different dot patterns according to the color indicated for a given area in the GIF image file. The name of the general technique used for creating the appropriate pixel pattern for each area of the image is called "dithering".

Dithering is defined as trembling. It is the way an artist blends colors with the tip of his brush to soften or smooth out the border between adjacent areas. On a hi-rez board, the brightness of each individual pixel in a given area can not be altered, but the proportion of lighted pixels to dark pixels can be controlled. The method of determining pixel area shading in this manner is called the dithering algorithm. The manner by which different algorithms alter the image must be seen rather than be described. I tried to describe them in the first review and, from that, I contend that viewing a GIF image is literally worth over a thousand words. There are more problems to be overcome, however, as we shall see.

The problems of size and proportion

While a monochrome television has no problem dealing with color, there is one problem that the television industry has had to deal with that makes it sound like the computer industry. There is an apparent economic payoff if you can keep your customers locked into a medium where you, the producer in the medium, can control the distribution of your product. While product quality, and not medium exclusivity, has proved to drive the marketplace, being able to lock the market up through an engineered incompatibility has always been an economically attractive proposal. It used to be called cornering the market. It has never been successful but it will always be tried. Amazingly, the eventual winner is the consumer because most incompatibilities have had to be improvements to have been marketable.

Old movies fit on a television screen very nicely because television has the same aspect ratio (width to height ratio) as the old movie screen standard. Movies went to color just as television started out. I believe that they went to color partially because television started out. So, the movie industry saw no threat to the television viewing of movies because they could only be seen in monochrome, even if the movie was in color. After all, the television standard originally lacked a provision for color.

When that fact changed, the movie industry took steps to prevent movies from being shown on television without some degradation in quality. Hence, Cinemascope, a different aspect ratio for movies versus television was born. Currently, high definition television promises to have a nearly Cinemascope aspect ratio. I wonder what is in store for the movies. Have they learned?

The relevance of the aspect ratio problem for the computer graphics industry is slightly different in origin but the same in practice. Every manufacturer adopted its own graphics standard, hoping that ITS standard would be THE standard. Even in the MS-DOS world, there are differing graphics standards.

Without laboring the point any further, how does the television industry handle the aspect ratio incompatibility? There are basically only three ways to do it. We have seen them all on our television screens from time to time. The most common solution is to crop the ends off of the wider aspect ratio image. The second most common solution is to compress the image in the direction wider direction until it fits the narrower medium. Lastly, the least common method (but my favorite) is to reduce the size of the entire image until its widest dimension fits within the narrower format.

Television usually uses the first option because the editor "knows" in advance what is of interest in the image and decides how much of each end of the wider image to sacrifice. However, when the entire width of the image must be shown at once, as with the credits, the entire image is distorted so that the cowboy on his horse looks like a tall space creature riding a giraffe. This is the most objectional form of editing. The third form of editing is admittedly "hard on the hardware" but it provides the most undistorted view of the original image. It is not used often because parts of the screen would be noticeably blank or dark and the picture tube would become unevenly burned in.

Gratefully, the last option is the method that GIF4MOD4 uses to render square pixel images, under the optional control of the user. The difference between my television and my computer is that the computer allows me a creative capability that my television lacks, (unless I were to start to make my own home movies). In the case of differing aspect ratios, I understand the problem, the solutions available, and the limitations of each. I hope you do also, now. Where we may tolerate the editing decisions of others in their broadcast television presentations, I think we would all like to make our own graphics decisions on our computers.

There are also problems with the size differences between different graphics standards. They are similar to the incompatibilities we have between our own Model I/III screen sizes and Model 4. Since we have all encountered these problems, I will not labor over them. Suffice it to say that GIF accommodates screen sizes up to 65535 by 65535 pixels. In my last review, I mistakenly stated that the limit for GIF (tm) was 640 by 480 pixels. That is simply the limit that GIF4MOD4 can accommodate due to our memory size limits.

Enter: GIF4MOD4 VERSION 2

Now that I have rambled on about the problems posed by computer graphics interchange, I have tried to relate them to similar problems in the television broadcast of feature films. This is because we have all seen, in broadcast

television, what has to be done to perform image transfers between incompatible media. The only way to see these solutions and their results on our hi-rez boards is to see GIF4MOD4 in action. There is one other way to do it if you don't have a hi-rez board, but I will cover that option later.

In my earlier review of GIF4MOD4, I had stated that GIF4MOD4 Version 1 had solved these similar problems in computer graphics Interchange, to the limits of our hardware. It does, in some respects. Then why is there a need for Version 2? Firstly, you may recall that I had a few criticisms about Version 1. Some were minor bugs which could be patched. Secondly, there were problems I had encountered that I thought, were unsolvable, (not so). Thirdly, there were apparently other problems that I didn't even encounter, which were solved in Version 2. Amazingly, virtually every criticism I made about Version 1 was addressed and every suggestion was implemented. Nobody can ask for more than everything. Yet, that is what we have in Version 2.

The installation instructions for a stock, single sided, two drive system appear very straight forward. If you do not have any other drive capacity, you will need to make a minimal system disk. Gratefully, I did not have to do this because I have external drives (and a hard disk). A standard memdisk, however, will provide you with all the speed you will need in most cases. Writing files of the processed image and/or for an interlaced image, to a disk file are exceptions. Disk capacity above the minimum requirement, however, is the most critical requirement. Otherwise the only concern is speed. A ramdisk would solve all questions of drive capacity and speed, immediately. It is a luxury but definitely not a requirement.

The most striking initial difference over Version 1 is the speed of Version 2. I was quite amazed to see Version 1 put ANY graphics image on my previously unused hi-rez screen, even if it took between 4 to 4 1/2 minutes to complete the process. When that time is reduced by Version 2 to under 1 minute, it seems as though I put an XLR8er board into my computer, although I have not. I am beginning to wonder what would happen if I DID. It actually took as little as 39 seconds for BATMAN/GIF, a 3K, 300 x 200 x 4 levels of gray CGA file, on a Gate Array Green Screen Model 4 with a standard memdisk and the factory seal intact, to process.

This speed difference is of increased practical importance when it is magnified by the additional dithering options available in Version 2. There are now a total of 7, plus some other options, but I will get to them soon. If you choose to dither an image, you will want to try every available dithering option. Most images require dithering and I recommend that you do try each option. Assuming that you allow the image to completely process each time, you are looking at a minimum total experimental processing time of over 1/2 hour per image.

With Version 2 and the GIF file of interest both in a standard memdisk, the same entire process can be completed in under ten minutes. That time can be reduced further if you quit processing before an image is complete and you are not satisfied with the way it looks, by hitting the <break> key. The program neatly allows this. You can reach your conclusion in seconds instead of minutes. However, your time increases again in Version 2 because a number of runtime options have been added which multiply your dithering menu choices by a factor of 4. You end up spending the same amount of time on an image you like. This is only because your viewing choices are greatly increased and you will want to take advantage of them all.

Color Controls

In Version 2, Mr. Slinkman has developed an error correction dithering algorithm that is adapted to the proper aspect ratio of our pixels. It uses a 1:2 aspect ratio rather than 1:1. This gives a result generally superior to algorithms designed for square pixels. Frank describes it better than I can since he created it. His descriptions of it can be found in the referenced articles at the end of this review. The dithering method is called the Slinkman Dither. I suspect that we will eventually see it in more places than on the TRS-80. Remember, we saw it in GIF4MOD4 Version 2 first.

Brightness, Contrast, Horizontal and Vertical

Other familiar processing options make GIF4MOD4 Version 2 seem more like a television set than a computer program. Actually, we can now control the processing of a graphics image as completely as we can control the image on a monochrome television. The options that effect image processing are: +b, +c, +r, and +s.

The +b option adds a noticeable amount of brightness to the processed image. Frank suggests that it be used when dumping graphics to the printer. I did not do any printer dumps because I could see the effect directly. It gives a pattern to a normally black appearing background but foreground images are brighter also.

The +c option increases the contrast of the entire image. Brighter areas become brighter still while darker areas will lose most or all patterning. An extreme case of this effect occurs if the "no dither" option is selected. All brightness values are divided right down the middle. This option, however, is less extreme and useful in conjunction with the other dithering options.

The +r option increases both the contrast and brightness of the entire image, effecting the brighter areas of the image most of all. The +r stands for exponential ramping of the colors. I have a qualitative idea of what it is doing from the terms but I am not sure what the process is actually doing. The effect is pleasant and useful. I know that an explanation like this is useless if you have not seen it, but it requires little explanation to detect if you are looking right at it.

The +s option is the opposite of the +c option. It decreases the contrast of the entire image. It is there if you need it.

The +d option, while simple to invoke and probably to program, is one of the most elegant parameters in the

program. It allows you to process GIF files, **even if you don't have a hi-rez board** to see them on. Why would you want to do this? Because you can later output the /HR files to your printer! This option gives you graphics capability in the absence of graphics hardware. You will need a utility to dump the file, but they are available in both commercial programs and in the public domain, as mentioned earlier. If your hi-rez board is not installed or inoperable, you have print utilities right on the included disk.

Frank has also written a program to print out GIF images, in very high resolution, on a Radio Shack DMP-2100, a HP DeskJet, and a C. Itoh 8510 - also known as the Apple Imagewriter). If you have any of these three printers ONLY, he will include the appropriate printer driver programs.

The +d option also allows you to create a printable file without having to load the editor portion of your draw/paint program to extract the file from your hi-rez board, even if you are using it. If you do not invoke it and your hi-rez board is not operating or present, the program will ask if you want to process to a disk file anyway. If not, it will quit. That is safe, clean, and neat.

The other optional parameters available are drive directives. They are :s and :d. They allow you to direct the source, temporary interlace, and output files between any valid drive(s). These are very necessary conveniences/options because they save the renumbering and write protection of drives. I never processed an interlaced file under Version 1 because I didn't have one. Without drive specification capability I might not have been able to do so. But no matter, it's there now.

You may remember my earlier mention of incompatible aspect ratios and their effects on image processing. I did not mention, however, that in digital images, aspect ratio incompatibility can occur at two different levels. There can be different ratios between the number of pixels in height and width of an image. This is the obvious level of incompatibility. The level that is not so obvious is that the pixels themselves can have different aspect ratios.

It's like building a square or rectangular patio with SQUARE bricks or building the same shape patio with RECTANGULAR bricks. If I told you to make a patio 200 bricks high and 200 bricks wide using standard rectangular red bricks, all placed in the same orientation, you might be surprised to discover that the patio would not be square, until you thought about it.

The graphic standards for many computer systems are based on the broadcast television screen aspect ratio of 4:3, but not all. Many also use rectangular pixels with an aspect ratio of 1:2. But, others work with square pixels having an aspect ratio of 1:1. Thus, two graphics standards with the same width to height ratios in number of pixels may have different overall aspect ratios due to the differing aspect ratios of the pixels they use. This is the other level of incompatibility between graphics standards.

Version 2 deals with both levels. It automatically scales the number of pixels in the width and height of the source file to the nearest simple ratio between the source and our hi-rez screens. If it can not do this exactly, it informs the user to expect a small degree of distortion. GIF (V87a), itself however, apparently does not have a descriptor for the aspect ratio of the pixels that originally composed the image. Since this information is not in the file, we must manually decide whether the image was originally created using [F]ull screen (the TRS-80 and most common standard) or [S]quare pixels. When appropriate, a menu is displayed requesting a decision. This decision can best be made by inspecting a version of the file, processed each way. As I stated earlier, I like the fact that the image is not cropped. Rather, it is displayed smaller than the full screen but entirely.

GIF (V87a) supports two different image formats and, therefore, so does GIF4MOD4 Version 2. The two file formats are Normal files versus Interlaced files. Normal files process from left to right and from top to bottom. Interlaced files load in the same direction, but only every eighth line gets processed, starting from the top. Then, every eighth line starting from line 4 gets processed. Then, every fourth line starting from line 2 is processed. Finally, every other line starting at line 1 completes the image.

The process is interesting to watch and it requires the creation of a temporary file. The reasons that interlaced files exist is unclear to me except that most interlaced files are large. Video scanning is often interlaced to reduce the bandwidth (basically, the amount of information) that the video system must support on a frame by frame basis, per unit of time. Since an interlaced file usually requires more time to process than a normal file, I am guessing that interlacing somehow allows for a greater processing capacity. After processing some interlaced files, I can tell you that they are not interlaced to take less time to process.

Bells and Whistles

The program, as it stands, is as complete and faithful to its function as it can be. Therefore, the following programs and files are convenience and adjunctive bonuses. For its purpose, each is just what is needed.

True interchange means that you can send as well as receive. Therefore, HR2GIF/CMD performs just the function you would expect. Give it a file with a valid extension, a source drive, a destination drive, and the choice of format you want: **[N]** ormal or **[I]** nterlaced.

HR2GIF/CMD will process the image file into a 640 x 240 x 2, Normal or Interlaced GIFfile. That is: 640 pixels wide by 240 pixels high by two colors (black and white). The syntax is: **HR2GIF/CMD fileid/ext :s :d**

For files smaller than this, such as Model III 512 x 192 files and image blocks from (partial image files) from some draw/paint programs, HR2GIF/CMD will automatically center the image on the Model 4 standard 640x240x2 file with your choice: **[B]** lack **[W]** hite background.

Again, we have the most elegant solution to screen aspect ratio incompatibility. It allows anyone receiving the file the greatest opportunity to work further with it.

APENDGIF/BAS is the next logical bonus that seems to fit, only after you discover it. I never thought of combining files in this manner. I would have expected to combine files on-screen with an editing paint/draw program. Instead, files can apparently be processed on top of each other, in sequence, forming an animated "slide show". I don't see how this phenomenon can be transferred to paper but it is quite effective on screen. Do not expect to be overwhelmed by the speed of animation.

Of all the features in this package, APENDGIF/BAS appears to have the least apparent utility. I have learned, however, that just because I don't see an immediate use for this feature, that someone else will not find use for it every day. Admittedly, if I had seen ABULLY/GIF in action the day I bought my Model 4, I would have bought it with a hi-rez graphics board right in it..... (*I did buy my Model 4, with the hi-rez graphics board right in it*).

Five JCL files are included in GIF4MOD4 Version 2. Four of them patch the GIF description of an image file to permit automatic processing of the file in the future. One allows the default drive for storage of the temporary interlace image file. There are ample warnings in the documentation that some of these files are irreversible and should be used sparingly. Personally, I would only use them on a backup of any image file I would patch.

Most of the time I found the options menus sparse enough that I didn't mind the single keystroke to get through them. I found that temporarily renaming GIF4MOD4/CMD TO G/CMD and renaming the image file I was working with, to a single letter such as BATMAN/GIF to B/GIF, made running the process faster than using a shell. This is because after a file was renamed, the program and file required less keystrokes to invoke. I used SHELL 2.0 to rename the files. That includes adding the optional parameters.

The five JCL files are as follow:

CGA2TRS/JCL: Changes screen height from 200 (CGA height) to 240 (TRS-80 height) and forces GIF4MOD4/CMD to treat CGA pixels as "square" instead of 5:6.

EGA2VGA/JCL: Changes screen height from 350 (EGA) to 480 (VGA) and forces GIF4MOD4/CMD to treat EGA pixels as "square" instead of 35:48.

FORCEFUL/JCL: Changes screen height from 200 (CGA height) to 201 and forces GIF4MOD4/CMD to treat CGA image as "full screen".

PIKDRIVE/JCL: Patches GIF4MOD4 to change the default target drive for /TMP files.

VGA2EGA/JCL: Changes screen height from 480 (VGA) to 350 (EGA). Use this file ONLY to reverse changes made to a GIF file by EGA2VGA/JCL.

Frank Slinkman did not have much to say about CHECKGIF/BAS but it was something I was curious about. I wanted to be able to find out what information a GIF file contains. In Version 1, little information about the parameters of the file was available. Version 2 gives more information about the files, but not about how many colors are in the file at runtime. That turns out not to be critical in practice but I still wanted to know. Additionally, I have concluded that if you start collecting and cataloging a large volume of image files in their original formats, an edited form of this program would be useful. The information that CHECKGIF/BAS extracts, can be LPRINTed or saved to a database, along with other information about the image file. Such a method can ease the task of organizing a practical graphics data base. Therefore I still like the program.

One bug, one fix, one criticism, one retraction, not bad.

One very minor bug persisted in GIF4MOD4 from Version 1 to Version 2. The [F] / [S] persisted on the screen during image processing. This had no effect on the processing of the program and was only a cosmetic bug. Frank didn't see it because he only has a Microlabs board and it shows up only on a Radio Shack board, which I have. He has already mailed out a patch to fix the bug, which I haven't applied yet, because it has not been that big a priority, if you know what I mean. I will do so after writing this review and I am sure that it will work. That ten line patch will update Version 2 to Version 2.1.

I mentioned GIF4MOD4 to a club member at one of our user groups and told him what it can do. He wanted it to do one thing that it doesn't do. He wanted it to do color separations by thresholds. He asked me if it can be done and I decided that it already does this when it does not dither. He wanted it to separate out the primary colors (color groups) so that they can be printed separately, in different color overlays.

I thought about that and mentioned the proposal to him at another meeting. He said that he thought about it also and decided that even if it could be done, it would not be practical because there is no means of registering the different overlays in a color xerographic copier, anyway. I asked him how often he needed a feature like that and he said that he never has yet. I asked him if it was worth pursuing and he said "No". That is the closest thing to a suggestion for improvement that I have encountered or generated. I'm satisfied and I think you will be also.

One last word

If I have made this review sound more like a review of a television system than a computer program, it is not by mistake. That is what the program will remind you of. Increasingly, the two technologies are starting to interrelate. GIF4MOD4 Version 2 allows us to tune into this phenomenon, to the limits of our systems.

References:
TRSTimes 3.1. Jan/Feb 1990
GIF4MOD4 Version 2 Documentation
COMPUTER NEWS 80 November 1989
BYTE September, 1989

HINTS & TIPS

DIGITIME REVISITED
for Model I, III & 4
By Carol Welcomb

Here are my versions of DIGITIME/BAS. The first, called BGCLOCK3/BAS, is for Model I/III. It includes a seconds display on the clock. With this program, I could place the last digit no further to the right than PRINT@59.

I worked on the Model 4 conversion to the point of thinking that I had completely lost my mind (which isn't actually saying much, I believe...). I actually figured out some things concerning the difference between Model III versus Model 4 video. However, despite the fact I can tell you what the screen won't do, I'm not certain if I can explain why the screens do what they do!

The next two programs are for Model 4. BGCLOCK4/BAS uses numbers I designed (smaller than DIGITIME/BAS) because I could not make the large numbers fit on the screen at the same time. The other, SMLCLOCK/BAS, is a demonstration of how far to the right the numbers can be placed on the screen. In this program, the last digit can be placed no further to the right than PRINT@(xx,40).

I am no fantastic programmer, but I have a compusion to figure out a dilemma, and this clock program in TRSTimes 2.6. was a true dilemma.

```
1 'BGCLOCK3/BAS
5 'by Carol L. Welcomb
10 CLEAR 1000
20 DEFSTR A-N,X,Z
30 M=CHR$(26)+STRING$(4,24)
40 X=CHR$(128):Z=CHR$(191)
50 A=STRING$(4,Z)
60 B=STRING$(2,X)+STRING$(2,Z)
70 C=Z+STRING$(2,X)+Z
80 D=STRING$(2,Z)+STRING$(2,X)
90 E=Z+STRING$(3,X)
100 F=STRING$(3,X)+Z
110 G=STRING$(3,Z)+X
120 H=X+STRING$(2,Z)+X
130 J(0)=A+M+A+M+C+M+A+M+A
140 J(1)=G+M+H+M+H+M+H+M+A
150 J(2)=A+M+B+M+A+M+D+M+A
160 J(3)=A+M+B+M+A+M+B+M+A
170 J(4)=C+M+C+M+A+M+F+M+F
180 J(5)=A+M+D+M+D+M+B+M+A
190 J(6)=A+M+E+M+A+M+C+M+A
200 J(7)=A+M+F+M+F+M+F+M+F
210 J(8)=A+M+C+M+A+M+C+M+A
220 J(9)=A+M+C+M+A+M+F+M+A
230 PRINT CHR$(15):CLS
240 N=Z
250 J=RIGHT$(TIME$,8):R1=VAL(MID$(J,1,1)):
R2=VAL(MID$(J,2,1)):S1=VAL(MID$(J,4,1)):
S2=VAL(MID$(J,5,1)):T1=VAL(MID$(J,7,1)):
T2=VAL(MID$(J,8,1))
260 PRINT@271,J(R1);:PRINT@276,J(R2);:
PRINT@345,N;:PRINT@473,N;:PRINT@283,J(S1);:
PRINT@288,J(S2);:PRINT@357,N;:PRINT@485,N;:
PRINT@295,J(T1);:PRINT@301,J(T2);
270 GOTO 250
```

```
1 'BGCLOCK4/BAS
2 'by Carol L. Welcomb
5 'based on "DIGITIME/BAS" in TRSTImes
10 CLEAR
20 DIM J(12)
30 DEFSTR A-N,X,Z
40 M=CHR$(26)+STRING$(4,24)
50 X=CHR$(128):Z=CHR$(191)
60 A=STRING$(4,Z)
70 B=STRING$(2,X)+STRING$(2,Z)
80 C=Z+STRING$(2,X)+Z
90 D=STRING$(2,Z)+STRING$(2,X)
100 E=Z+STRING$(3,X)
110 F=STRING$(3,X)+Z
120 G=STRING$(3,Z)+X
130 H=X+STRING$(2,Z)+X
140 J(0)=A+M+A+M+C+M+A+M+A
150 J(1)=G+M+H+M+H+M+H+M+A
160 J(2)=A+M+B+M+B+M+D+M+A
170 J(3)=A+M+B+M+A+M+B+M+A
180 J(4)=C+M+C+M+A+M+F+M+F
190 J(5)=A+M+D+M+D+M+B+M+A
200 J(6)=A+M+E+M+A+M+C+M+A
210 J(7)=A+M+F+M+F+M+F+M+F
220 J(8)=A+M+C+M+A+M+C+M+A
230 J(9)=A+M+C+M+A+M+F+M+A
240 PRINT CHR$(15):CLS
250 N=Z
260 J=RIGHT$(TIME$,8):R1=VAL(MID$(J,1,1)):
R2=VAL(MID$(J,2,1)):S1=VAL(MID$(J,4,1)):
S2=VAL(MID$(J,5,1)):T1=VAL(MID$(J,7,1)):
T2=VAL(MID$(J,8,1))
270 PRINT@(10,11),J(R1);:PRINT@(10,16),J(R2);:
PRINT@(11,21),N;:PRINT@(13,21),N;
```

```
280 PRINT@(10,23),J(S1);:PRINT@(10,28),J(S2);:
PRINT@(11,33),N;:PRINT@(13,33),N;:
PRINT@(10,35),J(T1);:PRINT@(10,40),J(T2);
290 GOTO 260
```

```
10 'SMLCLOCK/BAS
15 'by Carol L. Welcomb
20 'For Model 4, based on "DIGITIME/BAS" in TRSTimes
30 CLEAR
40 DEFSTR A-K,N,X,Y,Z
50 N=CHR$(26)+STRING$(4,24)
60 X=CHR$(128):Y=CHR$(131):Z=CHR$(191)
70 A=X+Y+X+X:B=X+X+X+Y:C=X+Z+X+X
80 D=X+X+X+Z:E=Y+Y+Y+Z:F=Z+Y+Y+Z
90 G=Z+X+X+Z:H=Z+Y+Y+Y:I=Y+Y+Y+Y
100 J(0)=F+N+G+N+I
110 J(1)=C+N+C+N+A
120 J(2)=E+N+H+N+I
130 J(3)=E+N+E+N+I
140 J(4)=G+N+E+N+B
150 J(5)=H+N+E+N+I
160 J(6)=H+N+F+N+I
170 J(7)=E+N+D+N+B
180 J(8)=F+N+F+N+I
190 J(9)=F+N+E+N+I
200 K=CHR$(176)
210 PRINT CHR$(15):CLS
220 J=RIGHT$(TIME$,8):L1=VAL(MID$(J,1,1)):
L2=VAL(MID$(J,2,1)):M1=VAL(MID$(J,4,1)):
M2=VAL(MID$(J,5,1)):O1=VAL(MID$(J,7,1)):
O2=VAL(MID$(J,8,1)) 230
PRINT@(12,19),J(L1);:PRINT@(12,24),J(L2);:
PRINT@(12,30),K;:PRINT@(13,30),K;
240 PRINT@(12,33),J(M1);:PRINT@(12,38),J(M2);:
PRINT@(12,44),K;:PRINT@(13,44),K;:
PRINT@(12,47),J(O1);:PRINT@(12,52),J(O2);
250 GOTO 220
```

HORIZONTAL SCROLLING
for Model I & III
By Lance Wolstrup

Here is a small subroutine that will put some flash in your Basic programs for Mod I & III. SCROLL/BAS will take the contents of the screen and smoothly scroll if off, either to the right or to the left. It's a fun way to clear the screen, and it certainly is an attention-getter.

SCROLL/BAS consists of two machine langauge routines. The first, scrolling the screen right, is found in the data statements in lines 10-13, and the second routine, which scrolls the screen left, is made up of the data statements in lines 15-19.

Line 20 uses a technique called 'string-packing'; that is, the data making up the ML routines are read into ML$. We are then sent to line 100, presumably the start of the user program.

Now, whenever the user program wishes to scroll the screen right, simply execute this line:

GOSUB 30:AA=USR0(0)

When the user program wants the screen to scroll left, use this code:

GOSUB 30:AA=USR1(0)

The subroutine in line 30 finds the actual current address of ML$ and stores that value in variable A.

Line 40 then defines the first routine (USR0) to start at the memory location pointed to by variable A. Line 45 defines the second routine (USR1) to start at A+57. Line 50 returns to the calling line.

Note the little programming trick: BOTH routines are stored in the same string. As a matter of fact, if I want to, I can store up to 10 ML routines in ML$ - as long as the total byte count of the routines does not exceed the length limit of the string.

Type in SCROLL/BAS and have fun with the new scrolling abilities.

```
1 'SCROLL/BAS
2 'subroutine - scrolls entire screen left or right
3 ';by Lance Wolstrup
4 '
10 DATA 6,64,197,6,63,33,62,60,17,63,60,197,229,6,16
11 DATA 197,229,1,1,0,237,176,225,17,64,0,25,229,209,19
12 DATA 193,16,238,225,193,229,209,43,16,227,6,15,17,64,0
13 DATA 33,0,60,54,32,25,16,251,193,16,202,201
15 DATA 6,64,197,6,63,221,33,1,60,221,229,225,229,209
16 DATA 27,197,229,6,16,197,229,1,1,0,237,176,225,17
17 DATA 64,0,25,229,209,27,193,16,238,225,193,229,209
18 DATA 35,16,227,6,16,17,62,0,221,229,225,25,19,19,54
19 DATA 32,25,16,251,193,16,195,201
20 FOR X=0 TO 120:READ A:ML$=ML$+CHR$(A):
NEXT:GOTO 100
30 A=PEEK(VARPTR(ML$)+2)*256+PEEK(VARPTR(ML$)+1):
IF A>32767 THEN A=A-65536
40 DEF USR0=A
45 DEF USR1=A+57
50 RETURN
100 'user program starts here - for example:
100 GOSUB 30:AA=USR0(0)
 OR
100 GOSUB 30:AA=USR1(0)
```

CHAOS INTO ORDER
for Model 4
By Robert Doerr

The PBS NOVA presentation on CHAOS inspired me to write the following program for Model 4.

```
5 'CHAOS/BAS
6 ' by Robert Doerr
7 '
10 DEFINT A-Z:DIM A(3),B(3):A(1)=0:B(1)=0:
A(2)=0:B(2)=79:A(3)=24:B(3)=RND(59)+10
20 Y=RND(23):X=RND(79):
IF X<(Y*B(3))/24 OR X>79-(Y/24)*(79-B(3)) THEN 20
ELSE CLS
50 U=RND(3):Y=(Y+A(U))/2:X=(X+B(U))/2:
PRINT@(Y-1,X),"*";:GOTO 50
```

UNPROTECT A PROTECTED BASIC PROGRAM

for Model 4

By Don Bergthold

One of the questions I get asked a lot is how one can get back into a BASIC program once it has been saved with the ,P option. Actually, it is very easy. Load the program back into BASIC, but don't run it. Then type:

```
SYSTEM"MEMORY (ADD=X'72CB',BYTE=0)"
```

You can now list, edit, or resave without the ,P option.

MULTIDOS 2.1. FOLLOW-UP

by David Welsh

Editor's note: This is an excerpt from a response to Jim King's letter, which was sent directly to AlphaBit Communications, Inc, and published in part in TRSTimes 3.1. under the heading: "WHAT'S WITH MULTIDOS 80/64 v.2.1."

BASIC TABS

The BASIC TAB works only if FORMS is set. I don't think it should work this way, but VERN (*Hester, author of MULTIDOS*) says the code that takes care of tabs depends on the FORMS code and, when FORMS are disabled, this code simply isn't executed. If you set FORMS on to any value, the tabs work just as they are supposed to.

RECOGNIZING DOUBLE SIDED DISKS

Vern told me he couldn't sqeeze in automatic recognition of 'doublesidedness' into the code, but CAT (L) or CAT (N) will read a a disk and properly and automatically CONFIG the drive in question for the disk. When in doubt, use CAT :d (N) or CAT :d (L) to find out, then the system will be CONFIGed correctly.

PROGRAMMING FUNCTION KEYS

If you are an assembly language programmer, you can cause the F keys to invoke a machine language routine at a certain address by putting the starting address in register DE and executing a restart 0 - (RST 0).

Location			Function
0	INPUT	USES	FUNCTION
(RST 0)	A=0	AF	Clear display
	A=1	AF	Set VIDEO to 40x10 and clear display
	A=2	AF	Set VIDEO to 50x18 and clear display
	A=3 B=0	AF	Set F1 thru F6 to default
	A=3 B=1	AF	Set F1 routine to @DE
	A=3 B=2	AF	Set F2 routine to @DE
	A=3 B=3	AF	Set F3 routine to @DE
	A=3 B=4	AF	Set F4 routine to @DE (RIGHT SHIFT F1)
	A=3 B=5	AF	Set F5 routine to @DE (RIGHT SHIFT F2)
	A=3 B=6	AF	Set F6 routine to @DE (RIGHT SHIFT F3)
	A=3 B>6	AF	Ignored
	A>3	AF	Ignored

The keyboard scan routine will execute the function, then return with the character in the A register. If a particular key input is also desired, then LD A with the character just before the last RET instruction. Function key commands/charactrers do NOT repeat.

Most of us just want to change the value of the character returned. You can do this by poking into memory. Below is a dump of the section of memory that returns characters when the F keys are pressed.

```
M0700H 06DD CBDB 6620 0BC0 3A40 38D6 04C8 2107
M0710H 07E5 DBEA E680 C8DB EBC9 79CD D207 CDDB
M0720H 07DB EAE6 4028 FA79 D3EB C93A 0238 FE1C
M0730H 2806 FE0D C03E 1801 3E38 3251 07CD AB44
M0740H C047 4F3A 5A00 F5CD 4C05 7EFE 2038 04FE
M0750H 8018 023E 2ECD 3B00 CD3B 0520 EA3E 0DCD
M0760H 3B00 CD00 0520 E0F1 CD77 05AF C911 1942
M0770H 0E3A 0603 1A36 2F34 D60A 30FB CE39 2377
M0780H 2305 C871 231D 18EC 111C 420E 2F18 E3C5
M0790H 4F06 10AF 2917 3803 B938 022C 9110 F5C1
M07A0H C921 2D40 31FE 42E5 21B6 07CD 6744 CD49
M07B0H 00D6 0D20 F9C9 496E 7365 7274 2053 5953
M07C0H 5445 4D20 3C45 4E54 4552 3E0D C955 00C9
M07D0H 5500 C933 00C9 3300 C93B 00C9 3B00 3E5C
M07E0H C93E 5DC9 3E5F C93E 7BC9 3E7C C93E 7DC9
M07F0H BA03 BA03 BA03 714C 8CE7 BA03 BA03 BA03
```

The bolded section is what we are interested in. The 3Es are LD A, instructions. The next byte is the value to return and the C9s are the return instruction. The one to change is the value between 3E and C9. In this case, I've changed the value of the F3 key by poking 95 decimal (5FH) into 07E5H.

I don't think this is discussed anywhere in the manual (although the portion above that tells how to cause a routine to execute in assembler is).

ASSEMBLY 101

Z-80 without tears

By Lance Wolstrup

In a previous issue of TRSTimes, this column presented TRSLABEL/CMD for mod I & III. The program, while relatively easy to write, proved very useful, and it generated a fair share of mail. "Translate it to Model 4", most of the letters said.

Fair enough, and as I can use a Model 4 version myself, let's do just that. The Model I/III version of TRSLABEL/CMD allowed the user to type a name, address, city-state & zip, country, and an additional line of information onto a formatted screen display. After the information was completed a mini editor allowed changing any of the entries. When everything was deemed correct, pressing the "0" key printed a mailing label. It had one additional option - pressing <CLEAR> would invoke preset information, such as the users return address, and pressing "0" would then print a label with that information.

TRSLABEL 4 - part 1

We will write the Model 4 version as described above - and add/modify a few things in the process. The mod I/III version spanned nine pages in the Sep/Oct 1989 issue and, as we add features, our new version will grow beyond that. This is more page space than can be allocated for any one article in TRSTimes, so I am forced to split the presentation into two parts. Part 1 follows below. It is the code and explanation of the main body of the program. Part 2, to be presented in the next issue, will contain all the subroutines, text messages and buffers, and their explanations.

Splitting the program into two installments creates the problem that the code in this issue will NOT work until the code from next issue is appended. You do not, however, have to wait to type in the listing. Fire up EDTASM (or other editor-assembler) and enter this portion of the code. When done, save the it to disk (in EDTASM use: **W TRSLABL4** <ENTER>). Do not attempt to assemble the code at this time, rather store the disk away until next issue.

Having gotten the caveat behind us, let's go on and make a list of the approximate program flow:

1. display name of the program & copyright information.
2. format the screen so all input prompts are visible.
3. draw underline characters at current input, showing the maximum length of the input.
4. allow user input.
5. go to editor where the information at any of the input fields can be changed.
6. at editor prompt - also supply prompts for QUIT, NEW LABEL and PRINT.

```
00050 ;TRSLABL4/SRC
00060 ;for Model 4 - TRSDOS 6.x/LS-DOS 6
00070 ;(c) 1989 Lance Wolstrup
00080 ;
00090 ;
00100          ORG    3000H
00110 START    CALL   CLS              ;erase screen
00120 ;
00130 ;change certain low memory locations to:
00140 ;force lower case - 74H
00150 ;force special characters - 0B94H
00160 ;turn cursor off - 0B97H
00170 ;change cursor to chr$(176) - 0B98H
00180 ;
00190          LD     A,(74H)          ;get Kflag$
00200          AND    223              ;reset bit 5
00210          LD     (74H),A          ;to force lower case
00220          LD     A,(0B94H)        ;set bit 3 of
00230          OR     8                ;0B94H to force
00240          LD     (0B94H),A        ;special characters
00250          XOR    A                ;a=0
00260          LD     (0B97H),A        ;make cursor invisible
00270          LD     A,176            ;change cursor to
00280          LD     (0B98H),A        ;chr$(176)
00290 ;
00300 ;display program name and copyright notice
00310 ;
00320          LD     HL,0000H         ;cursor @(0,0)
00330          CALL   LOCATE
00340          LD     HL,MSG1          ;display at cursor
00350          CALL   DSPLY
00360          LD     HL,0023H         ;cursor  @(0,35)
00370          CALL   LOCATE
00380          LD     HL,MSG2          ;display at cursor
00390          CALL   DSPLY
00400          LD     HL,003EH         ;cursor @(0,62)
00410          CALL   LOCATE
00420          LD     HL,MSG3          ;display at cursor
00430          CALL   DSPLY
00440          LD     HL,0113H         ;cursor @(1,19)
00450          CALL   LOCATE
00460          LD     HL,MSG4          ;display at cursor
00470          CALL   DSPLY
00480 ;
00490 ;draw solid line across line 2
00500 ;
00510          LD     HL,0200H         ;cursor  @(2,0)
00520          CALL   LOCATE
00530          LD     BC,5083H         ;b=counter,c=chr$(131)
00540 MLOOP    CALL   DSP              ;print string$(80,131)
00550          DJNZ   MLOOP            ;continue until done
00560 ;
00570 ;set up prompts for Name, Address, City, State & Zip,
```

a. QUIT will exit to DOS.
b. NEW LABEL will erase input fields and start over.
c. PRINT will go on to the print module.

7. print module will ask how many labels to print.
8. print the labels and return to the editor prompt.
9. at any time during the program the preset label can be invoked by pressing <SHIFT-CLEAR>.

Though points 1 through 9 pretty well explains the idea of the program, there are a few more things I ought to point out before we begin. Since Model 4 has a CONTROL key, I thought it might be fun to use it for some of the functions.

CTRL-N pressed at any time (except during print) will erase the information typed at the field prompts and let you start with a blank label. CTRL-Q pressed at any time (except during print) will exit directly to DOS. CTRL-Q pressed during print will exit the print routine and return to the editor prompt. CTRL-P pressed at any time (except during print) will go directly to the print routine.

By allowing direct access to the print routine from the input routine, you should be aware that it is possible to type one or more lines of new information while retaining the information stored at the unused lines from the previous label. For example - having just printed a label, you now want to print another where the bottom three lines are identical to the current label. Press <CTRL-N> for a new label and type the new information for the two top lines. Then press <CTRL-P> to enter the print module.

This was originally a programming error on my part, but the more I thought about it - the more I liked it, so it is now considered a "feature".

Now let's get down to the actual code. We begin by calling the CLS subroutine to clear the screen. Then we go on to change some strategic low memory locations for the duration of the program.

The byte at 74H (TRSDOS 6.2. and LS-DOS 6.3) is called KFLAG$. It contains information about the keyboard. Bit 5 determines the case status: OFF=lower case, and ON=upper case. Since I prefer to be in lower case, lines 190-210 copies the contents of KFLAG$ into register A where bit 5 is turned off by ANDing 223. This new value is then stored at KFLAG$ - and we are in lower case.

Lines 220-240 deal with memory location 0B94H. Bit 3 of this location acts as a switch between space compression mode and special character mode: OFF=space compression mode, and ON=special character mode. I wish to use the pointing hand (ASCII characters 244, 245 and 246) as part of the program name heading, so I must make sure that the special character mode is invoked. By copying the byte at 0B94H into register A, then ORing it with 8, bit 3 is turned on. The new value is then copied back to 0B94H, and special character mode is active.

```
00580 ;Country, and Other
00590 ;
00600 START1  LD      HL,0400H    ;cursor @(4,0)
00610         CALL    LOCATE
00620         LD      C,31        ;erase from cursor
00630         CALL    DSP         ;to end of screen
00640         LD      L,10H       ;cursor @(4,16)
00650         CALL    LOCATE
00660         LD      HL,NAME     ;display at cursor
00670         CALL    DSPLY
00680         LD      HL,050DH    ;cursor @(5,13)
00690         CALL    LOCATE
00700         LD      HL,ADDRS    ;display at cursor
00710         CALL    DSPLY
00720         LD      HL,0603H    ;cursor @(6,3)
00730         CALL    LOCATE
00740         LD      HL,CSZ      ;display at cursor
00750         CALL    DSPLY
00760         LD      HL,070DH    ;cursor @(7,13)
00770         CALL    LOCATE
00780         LD      HL,CNTRY    ;display at cursor
00790         CALL    DSPLY
00800         LD      HL,080FH    ;cursor @(8,15)
00810         CALL    LOCATE
00820         LD      HL,OTHER    ;display at cursor
00830         CALL    DSPLY
00840 ;
00850 ;begin input to the label
00860 ;
00870         CALL    INPUT1      ;user types name
00880         CALL    INPUT2      ;user types address
00890         CALL    INPUT3      ;user types city,state
00900         CALL    INPUT4      ;user types country
00910         CALL    INPUT5      ;user types other
00920 ;
00930 ;erase field headings
00940 ;
00950         LD      HL,0400H    ;cursor @(4,0)
00960         CALL    LOCATE
00970         LD      E,22        ;e=number of chrs
00980         CALL    ERASE       ;to erase per line
00990 ;
01000 ;enter edit mode
01010 ;
01020 EDIT    LD      HL,0A00H    ;cursor  @(10,0)
01030         CALL    LOCATE
01040         LD      C,31        ;erase from cursor
01050         CALL    DSP         ;to end of screen
01060         LD      HL,EDMSG    ;display at cursor
01070         CALL    DSPLY
01080 EDIT0   LD      B,1         ;only need 1 keystroke
01090         LD      IX,EDBUF    ;point ix to buffer
01100         CALL    INKEY
01110         LD      A,(EDBUF)   ;get chr
01120         CP      36H         ;is it "6" ?
01130         JR      NC,EDEXIT   ;no-than 5-not allowed
01140         CP      31H         ;is it "1" ?
01150         JR      C,EDEXIT    ;no-smaller-not allowed
01160         PUSH    AF          ;save keystroke
01170         LD      HL,0A00H    ;cursor @(10,0)
01180         CALL    LOCATE
```

Lines 250-260 store zero in memory location 0B97H, thus turning off the cursor.

Finally, memory location 0B98H holds the value of the cursor character (normally 95). By storing 176 there we change the cursor character to CHR$(176).

Now, I know, some of you are asking where the heck this information is written in the manual. IT ISN'T. As a matter of fact, a great many pains were taken by the authors of the DOS to avoid giving this information out. However, since I don't expect to see a TRSDOS 6.4. in the near future, I am not worried about using some absolute memory locations instead of SVC's. *(for a more comprehensive list of low memory locations refer to PEEKING & POKING Model 4 in TRSTimes 1.1. and 1.2.)*

Lines 320-470 display the program name and copyright on screen line 0, as well as a short description of the program on screen line 1.

Lines 510-550 position the cursor at (2,0) and then proceed to draw a solid line across screen line 2. Note that line 530 LoaDs register BC with 5083H. That is the exact same thing as LD B,50H (80 decimal) and LD C, 83H (131 decimal).

Lines 600-830 display the input prompts.

Lines 870-910 take care of the actual user inputs by calling individual subroutines for each input.

Returning from the last input, lines 950-980 erase the input prompts. We position the cursor at (4,0), store the value 22 in register E, and call the erase subroutine where 22 characters from the cursor position will be erased - for five lines. The erase subroutine will also place the numbers 1 to 5 in front of the user input for identification in the edit routine.

Line 1020-1070 begin the edit routine by positioning the cursor at (10,0), then erasing from there to the end of the screen. We then display the edit prompt.

Lines 1080-1090 set up needed parameter for the inkey subroutine. Before entering the routine, register B must contain the maximum number of characters allowed (In this case only one is allowed) and register IX must point to a buffer where the keystroke(s) will be stored.

Line 1100 calls the Inkey subroutine. This routine is the heart of the program and will be explained in detail in the next installment (sorry about that!)

Line 1110 retrieves the keystroke from the buffer and stores it register A.

Lines 1120-1150 send any keystroke smaller than "1" or larger than "5" to the edexit routine, thereby rejecting anything other than "1", "2", "3", "4", or "5".

If we get to line 1210, we have obviously chosen a valid field to edit, so we push the keystroke onto the stack and proceed to erase the the edit prompt (lines 1170-1200).

Line 1210 retrieves the keystroke, and we can then test which was pressed, followed by a jump to the appropriate routine (lines 1220-1280).

```
01190          LD      C,31        ;erase from cursor to
01200          CALL    DSP         ;end of screen
01210          POP     AF          ;restore keystroke
01220          JR      Z,EDIT1     ;jump if "1"
01230          CP      32H         ;is it "2" ?
01240          JR      Z,EDIT2     ;jump if "2"
01250          CP      33H         ;is it "3" ?
01260          JR      Z,EDIT3     ;jump if "3"
01270          CP      34H         ;is it "4" ?
01280          JR      Z,EDIT4     ;jump if "4"
01290 EDIT5    CALL    INPUT5      ;must be "5"-get input
01300          JR      EDIT        ;and return to edit
01310 EDIT4    CALL    INPUT4      ;it is "4" - get input
01320          JR      EDIT            ;and return to edit
01330 EDIT3    CALL    INPUT3      ;it is "3" - get input
01340          JR      EDIT        ;and return to edit
01350 EDIT2    CALL    INPUT2      ;it is "2" - get input
01360          JR      EDIT        ;and return to edit
01370 EDIT1    CALL    INPUT1      ;it is "1" - get input
01380          JR      EDIT        ;and return to edit
01390 EDEXIT   LD      HL,0A4EH    ;cursor @(10,78)
01400          CALL    LOCATE
01410          LD      C,32        ;erase the character
01420          CALL    DSP         ;at cursor position
01430          JR      EDIT0       ;and ret to edit prompt
01440 ;
01450 ;enter print mode
01460 ;
01470 PRINT0   LD      HL,0A00H    ;cursor @(10,0)
01480          CALL    LOCATE
01490          LD      C,31        ;erase from cursor to
01500          CALL    DSP         ;end of screen
01510          LD      L,10H       ;cursor @(10,16)
01520          CALL    LOCATE
01530          LD      HL,HOWMNY   ;display at cursor
01540          CALL    DSPLY
01550 PRINT1   LD      HL,0A3FH    ;cursor @(10,63)
01560          CALL    LOCATE
01570          LD      BC,0320H    ;b=loop counter, c=chr
01580 PLOOP    CALL    DSP         ;erase 3 input chrs
01590          DJNZ    PLOOP       ;continue until done
01600          LD      B,3         ;b=max chrs
01610          LD      IX,NUMBUF   ;point ix to buffer
01620          CALL    INKEY
01630          LD      HL,NUMBUF   ;point hl to buffer
01640          CALL    TSTCHR      ;check if input is valid
01650          JR      NZ,PRINT1   ;bad input - redo
01660          LD      A,96        ;@dechex
01670          RST     40          ;# of labels in bc
01680          LD      A,C         ;check low byte
01690          OR      A           ;is it 0 ?
01700          JR      NZ,PRINT2   ;no - go print
01710          LD      A,B         ;check high byte
01720          OR      A           ;is it 0 ?
01730          JP      Z,EDIT      ;yes - return to edit
01740 PRINT2   PUSH    BC          ;save loop counter
01750          LD      HL,NABUF    ;point to name buffer
01760          CALL    PRINT       ;print it
01770          LD      HL,ADBUF    ;point to address buffer
01780          CALL    PRINT       ;print it
01790          LD      HL,CSBUF    ;point to city-s buffer
```

Line 1290 is entered by default. Since the keystroke is 1, 2, 3, 4, or 5, and we haven't jumped to one of the other routines already, the keystroke must be "5". The input5 routine is called and, upon return, we go back to the beginning of the edit routine (line 1300).

Lines 1310-1380 are identical to the above routine, except they cover keystrokes "4", "3", "2", and "1".

The edexit routine at line 1390 is entered only when the keystroke is not "1". "2", "3", "4' or "5". This means an invalid key has been pressed, so we simply erase the keystroke displayed on the screen and go back to the prompt.

Lines 1470-2000 is the actual print routine. You can only get here by pressing <CTRL-P> from the inkey routine.

Lines 1470-1540 positions the cursor at (10,0), then erases from there to the end of the screen. Next the cursor is positioned at (10,16) and the prompt asking how many labels to print is displayed.

Starting at PRINT1 (line 1550) the cursor is placed at (10,63) and a loop erases the three input positions at (10,63 - 10,64 and 10,65). This is done because, if an incorrect input to the 'number of labels' prompt occurs, we are sent back to the PRINT1 routine.

Line 1600 loads the maximum number of input characters into register B. Then register IX is loaded with the address of the NUMBUF buffer, and the INKEY routine is called.

Coming back from the INKEY routine (line 1630) we need to know if the input is valid. Thus, we load register HL with the address of the NUMBUF buffer and proceed to the TSTCHR subroutine where each character is tested. If an illegal character is found, the routine returns with the NZ flag set. This condition is tested in line 1650, and if NZ is set we are returned to the PRINT1 routine for a new input.

If the Z flag is set, then the input is legal and we must now take the input and convert it into a numeric. Lines 1660-1670 perfom that task by invoking the @dechex SVC, which takes the ascii input stored in the buffer pointed to by HL and converts it to a numeric which it stores in register BC.

As a way of escaping the print routine at this point, the user may type 0 to the 'how many labels' prompt and the program will branch to the edit routine. To accomplish that we must test the value of register BC. If the value is 0 we will then know to branch to the edit routine. Line 1680 copies the low byte into A and line 1690 tests for 0. If it is not 0, there is no need to test the high byte and we can go on and begin printing labels. If, on the other hand, we do find a 0 in the low byte, we must test the high byte also (lines 1710-1720). If 0 is found here, then 0 was entered from the keyboard and we go to edit. If a non-zero value is found we go on to line 1740 where we immediately save the number of labels desired onto the stack. It will be used as a loop counter.

Lines 1750-1860 print the five fields and, since a label has six lines, an empty sixth line is printed, thus bringing the next label to the top of the print head.

Lines 1870-1880 invoke the @kbd SVC which checks for a keystroke but, unlike the @key SVC, goes on if no keystroke is found.

Lines 1890-1900 check if CTRL-Q was pressed. If so, the printing will abort and the program flow branches to the edit routine. If not, we go on.

Lines 1910-2000 checks how many labels we have left to print. First we pop the loop counter back into BC. Then, as we have already printed one label, we subtract one from the count. Now we need to see if the low byte is 0 (lines 1930-1940). If so, we need to check the high byte. If not, we go back and print another label.

The high byte is checked ONLY if the low byte is 0, as we need to know if we have come to the end of printing the labels. Lines 1960-1970 check the high byte and, if it is 0 we quit printing by going to the edit routine. If not 0, we subtract one from the high byte and go back and print more labels.

Lines 2020-2090 contain the QUIT routine. This is also accessible only from the INKEY routine. Here we first erase the screen (line 2020), then we reset the cursor to the normal underline character (lines 2030-2040) and turn the cursor on (lines 2050-2060). Finally the special characters are replaced with space compression (lines 2070-2090) and we return to DOS (line 2100).

Next issue will bring part 2 of TRSLABL4 - so until then, keep practicing.

```
01800        CALL   PRINT      ;print it
01810        LD     HL,CNBUF   ;point to country buffer
01820        CALL   PRINT      ;print it
01830        LD     HL,OTBUF   ;point to other buffer
01840        CALL   PRINT      ;print it
01850        LD     C,13       ;we need cr
01860        CALL   PRT        ;print cr
01870        LD     A,8        ;@kbd
01880        RST    40
01890        CP     17         ;was ctrl-q pressed ?
01900        JP     Z,EDIT     ;yes - stop printing
01910        POP    BC         ;restore loop counter
01920        DEC    C          ;decrement low byte
01930        LD     A,C        ;check low byte
01940        OR     A          ;is it 0 ?
01950        JR     NZ,PRINT2  ;no - print more labels
01960        LD     A,B        ;yes - check high byte
01970        OR     A          ;is high byte 0 ?
01980        JP     Z,EDIT     ;yes - end of labels
01990        DEC    B          ;decrement high byte
02000        JR     PRINT2     ;go print more labels
02010 ;
02020 QUIT   CALL   CLS        ;erase screen
02030        LD     A,95       ;reset cursor to
02040        LD     (0B98H),A  ;normal chr
02050        LD     A,32       ;make cursor visible
02060        LD     (0B97H),A
02070        LD     A,(0B94H)  ;get Kflag$
02080        AND    247        ;reset bit 3
02090        LD     (0B94H),A  ;now space compress.
02100 ;      RET               ;back to dos
```

A LITTLE HARD DISK PROBLEM part 2

By Roy Beck

Since my last essay on installing CP/M on my VR DATA Hard Disk III, I have been diligently working at unraveling the internal structure of the HD driver program from Montezuma Micro which is "close", but not right for my 5 Meg HD. My wife says I have been too diligent, but once I really got into it, it was hard to set aside for such trivialities as income taxes, etc!

In the last chapter, I listed some areas where I expected to make changes. As I made progress into the driver, I discovered many more items which must be examined and/or revised, so at this point I will recite my most recent list:

1. Name and size in the sign-on message.
2. Head count.
3. Track count.
4. Reduce write current track.
5. Precompensation track.
6. Step rate.
7. Directory track.
8. Abandon diagnostic track reservation if present.
9. Drive partitioning tables.
10. Table of constants to be installed in DPB.

As you can see, there are many factors to take into consideration, more than I had first imagined; and I am not even sure the list is complete at this point, as this chapter is really only a progress report.

I had originally assumed, in my naivete, that there really couldn't be much involved in disassembling only 3.5 K of code, much of which was obviously tables and messages. I now plead guilty to gross overoptimism! But having gotten this far, I am sure not going to give up on it.

Actually, I have now learned more than I ever wanted to know about the internals of CP/M's BIOS, but like medicine in childhood, you take it because it's "good for you". That's the way I feel about some of this code. I am following author Jesse Bob Overholt (JBO) along in the code as it executes, and at times I am totally puzzled, some times a little bit puzzled, and then occasionally the light dawns like a beautiful sunrise!

One of the areas which has been educational has been the extensive checking required to prevent wrong input by the user. For example, only 4 HD partitions are permitted. When writing the code, you must verify the user doesn't ask for 5 partitions. Another item is that there MUST be a drive A in CP/M. Without it, CP/M cannot function. JBO checks to insure you did install drive A somewhere in the system.

While I am following JBO's code, I am sometimes forced to divert into the world of SASI/SCSI to learn the protocols of handling that bus. (*It is interesting to note that the Macintosh world uses a SCSI port to handle their HD's, also*). While I wonder at times why they do things the way they do, I am forced to admit it all works. Fortunately, there are only a few SASI commands actually needed, which simplifies things. The commands in use appear to be limited to the following:

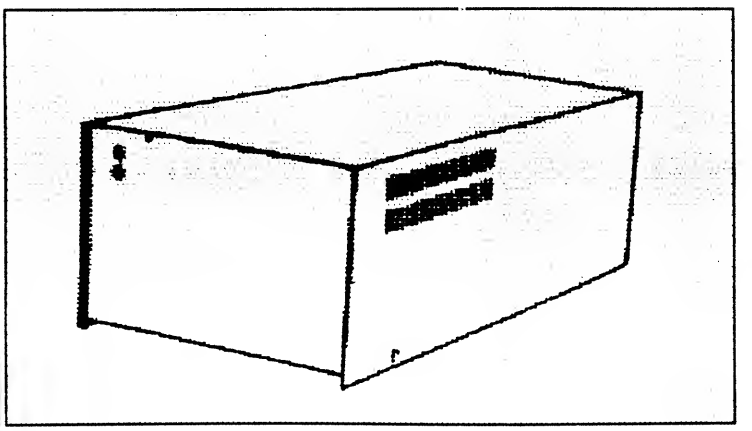
00H Test if drive is ready.
01H Restore heads to track 000
04H Read Sector
08H Write Sector
0CH Initialize controller
0FH Load sector buffer in HDC

An aspect of CP/M to which I had previously paid little attention and therefore didn't appreciate is its extensive use of tables. Most of them are essential design features of CP/M; but I think JBO added at least a couple on his own to facilitate his driver development. The great virtue of a table is that it forces you to organize material in an orderly way, and given the structure of the Z-80, the IX and IY registers can make effective use of the tables to simplify the code. Some of the tables I have found and studied include:

SPB System Parameter Block
DPH Drive Parameter Header
DPB Drive Parameter Block
DCB Drive Control Block

The SPB is sort of an overall roadmap to where things are in CP/M.

There is one DPH for each logical drive in CP/M, and this table includes the addresses of some other tables required by CP/M.



One of these is the Skew Translation Table, which contains the disk interleaving information. In CP/M, the sectors are usually formatted from 1 to whatever in numerical order along the track. But since the CPU usually cannot read them in successive order as they fly past the head, due to the need for time to digest the data between sectors, the skew translation table provides the necessary interleaving to read every 2nd or every 3rd or whatever sector as the disk rotates. Each successive entry in the table contains a physical sector number corresponding with the logical sector number which is denoted by its position in the table. Apparently interleaving is handled internally in the Xebec HDC, and therefore the skew table is not skewed; it is simply in order from 1 to 32.

Also located in the DPH is the address of the DPB which contains pointers, constants, etc, necessary to work out the block assignments for use in the directory structure.

JBO added another data structure called the DCB, which is used in LSDOS and DOSPLUS, and which he figured would be very valuable. In order to be able to find the DCB when needed, he grafted an extension onto the DPB mentioned above and then located an address pointer there which locates the DCB. By the way, if you review the structure of LSDOS or DOSPLUS, the table identified there as DCT is the same as JBO's DCB. That is to say, the concept is the same, although the detailed contents differ from one DOS to another.

Because JBO doesn't allow the user to assign his own head and track offsets vs logical drives, he had to predefine all the constants in some more tables. I don't know what he called these tables, but functionally the first one consists of a set of pointers for each partition. There is one of these tables for each possible configuration of the drive. In this case, he allowed for 1, 2, 3, and 4 partitions, therefore there are 4 partition tables. In addition, as he allowed for different sized partitions, he had to adjust the directory block size to suit the partition sizes. This in turn brought about the need for 3 more tables, corresponding to block sizes of 16K, 8K and 4K, respectively. These are plugged into the DPB's at the appropriate time.

Another of the tables he had to generate for use during installation of the driver, but could abandon later include a table of pointers to absolute addresses within his driver image which had to be modified after the final location of the driver is established. (This varies because the actual size of the BIOS may vary at the user's discretion, and JBO could not predict exactly where the driver would end up). Other tables include the table of flawed tracks, and copies of some of the other tables. After all, he cannot patch the DOS piecemeal while still executing it! Instead, he makes copies of all the tables to be patched, but does not implement the patched versions until everything checks out OK. Then in one fell swoop, he block-moves the patched tables over the original CP/M tables, and presto the HD is now in use!

I haven't gotten that far along yet, I am just at the flawed track recording stage now.

Time spent sleeping, drinking coffee, etc.

Since the last increment of this epistle to the faithful, I have found most of what I started out to discover. I have answers to all of the items listed at the beginning of this Part Two. In a couple of cases, the questions have gone away.

Item 7 referred to the directory track. In CP/M, the directory always begins in Allocation Block 0, which places it in the first track above the system track(s). In our case, two tracks are reserved for the system, and so the directory begins at relative track 2 of the logical drive. Since I intend to assign by head offset, that will place the directory on physical track 2 of each head.

Item 8 of my list referred to the diagnostic track reserved by Radio Shack on RS drives. Since this VR DATA drive is not by RS, and the driver I am working with is also not by RS, I am not sure there is a diagnostic track. I will attempt to search for it later, and ignore it for now.

At this point I should explain the drive partitioning I intend to apply. First, I am dealing with drive 1 of a two bubble system, with the bubbles numbered 0 and 1. (I have TRSDOS 6.3 on drive 0). I intend to partition drive 1 by head offset, thus minimizing the risk of head crashes. If one head crashes, the other 3 heads may remain useable, at least long enough to salvage the files on those other heads. Since I also want to put LDOS on this drive 1, I will restrict CP/M to just one head of the drive, arbitrarily head 3, the last one of the four.

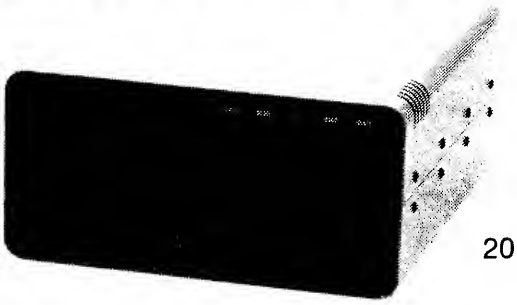
The Montezuma Micro driver I am modifying is arranged to allow partitioning by head offset, and optionally 1, 2, 3 or 4 logical partitions. In order to limit the drive to head 3 only for CP/M, I will tell the HD driver to setup for 4 logical partitions, one per head, but only assign one of the heads at this time. The other 3 heads will remain available for other purposes. (I know the driver will accommodate me in this manner, because I did a similar thing in configuring my 35 Meg HD at home. *See TRSTimes 2.5. - Sep/Oct 1989.*)

I will place Drive A on head 3 of the hard disk, and since I am operating on a 4P with only two floppies, I will only assign two of the four possible floppy drives initially.

To be continued.....

PERIPHERALS SALE!

Model 3/4 Hard Disk Drives



- Lowest Prices Ever
- Brand New Units
- FCC Class B Certified
- Complete with Cables
- Complete with Software
- Money Back Guarantee
- High Performance
- Reliable
- Thousands in Use

As Low As **$289** 5MB, 80ms

20MB, 65ms.. only **$449** 40MB, 40ms (28ms optional).. **$559**

Faster drives available at extra cost.
Add $22 packing and shipping in a custom made foam carton.
Brushed stainless steel case available. Add $20.

Software and cables included.

Aerocomp leads the way with lower prices for our loyal TRS-80 friends once again. Aerocomp drives are a TRS-80 standard and are available in three sizes. These are not uncertified kits, but new and complete units ready to run. All models include **brand new Seagate drives,** not some used drive or one that has been refurbished or from a second rate manufacturer's boneyard (Tandon, Miniscribe, etc.).

These external hard drives are FCC Class B Certified as required by law. Aerocomp hard drives are an established product and have survived the test of time. Thousands of satisfied Aerocomp hard drive customers have proven these products a solid value for their owners. A secondary hard drive can be added at any time you desire. Larger drives can be installed in your original case, thereby protecting your investment. The hard drive itself can even be transferred to an MS-DOS compatible computer if that is in your future.

Our units are complete with a 6' interface cable and the TRSDOS, LDOS or CP/M software driver of your choice at no additional cost.

Aerocomp provides all the little things that are so important for a long, trouble-free life: continuous-duty switching power supplies, filtered forced-air ventilation; effective EMI filtration, solid steel construction; five front panel indicator lights (Power-Ready-Read-Write-Select); built-in diagnostics, and gold plated connectors.

Probably the most important thing of all is our 30 day money back guarantee. If, for any reason, you are not satisfied with the drive, we'll refund the entire purchase price, less the shipping charges. The warranty is for one full year and includes all parts and labor.

Save Now On Our
LOW COST FLOPPYS

Aerocomp has been supplying quality disk drives at low prices since 1980. All drives are half-height and are new—not factory blems, seconds, close-outs or a defunct manufacturer's surplus (MPI, Qume, Tandon, etc.). We offer just about every combination of internal and external floppy configuration plus the proper cables to connect everything together. We appreciate your business and will do our very best to support you. If, for any reason, you aren't happy with your hardware selection, we'll cheerfully refund the entire purchase price, less shipping. Order yours today! All items (except software) have a one year parts and labor warranty.

BARE DRIVES

360K 5.25" TEAC 55B bare drive 40tk	$79
720K 5.25" TEAC 55F bare drive 80tk	109
1.2M 5.25" TEAC 55G bare drive 80tk	85
360K 3.50" TEAC 35B bare drive 40tk	59
720K 3.50" TEAC 35F bare drive 80tk	69
1.4M 3.50" TEAC 35H bare drive 80tk	69

DRIVE-POWER SUPPLY COMBINATIONS
(Includes gold plated extender)

1-TEAC 35B 360K dual enclosure	$129
1-TEAC 35F 720K dual enclosure	169
1-TEAC 55B 360K dual enclosure	149
1-TEAC 55F 720K dual enclosure	$149

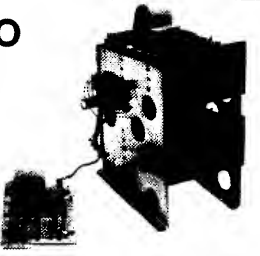
Add $10 for brushed stainless steel cover.

CABLES · CASES · DOS

IBM ext floppy cable (drives C/D)	$39
TRS-80 2-drive floppy cable	24
TRS-80 4-drive floppy cable	34
6' floppy ext cable gold contacts	12
3½" case, power supply w/o ext	49
5¼" case, power supply w/o ext	59
TRS-80 Model 4 CP/M (Montezver.)	69

Add $10 for brushed stainless steel cover.
Add $4 shipping for singles, $6 for duals.

ADD DISK DRIVES TO YOUR MODEL 3/4!



New Low Price! Complete System Less Drives, DOS

Now Only **$99** Save!

Convert your cassette Model 3 or 4 to fast disk operation with one of our easy-to-install kits. Complete instructions are provided. All you need is a screwdriver and a pair of pliers. Our own advanced controller, 100% compatible with the original, plated steel mounting towers with RFI shield and all cables and hardware included. Select your drives from the other column and call us, toll-free, to place your order. If, for any reason, you don't like the kit, we'll refund the entire purchase price, less shipping. Order yours today!

Disk Controller only $49

RS-232 Board complete $49

Add $5 shipping. One year parts and labor warranty.

DOUBLE DENSITY CONTROLLER

Now Only **$49** Add $4 Shipping

80% more disk capacity is what you get when you add our DDC to your TRS-80 Model 1. This controller has withstood the test of time. All the others are gone, yet the Aerocomp DDC endures. Why? Because it has proven itself as the only way to achieve reliable floppy disk operation on the Model 1. Requires the Radio Shack Expansion Interface and software driver. All DOS (except TRSDOS) have the necessary double density driver. If, for any reason, you don't like the DDC, we'll refund the entire purchase price, less shipping. Order yours today! One year parts and labor warranty.

OUT THEY GO!

ENJOY INCREDIBLE SAVINGS NOW ON SOFTWARE, BOOKS AND MANUALS...WHILE THEY LAST!

CP/M® SOFTWARE

Twist & Shout	$6

CP/M BOOKS & MANUALS

Inside CP/M, by Cortesi (book)	$2
CP/M System Prog. Manual, Model 4	5
Monte's Mail (newsltr), Vol 1 #1, Vol 2 #1	
Specify Volume ..... Each Volume	1

TRS-80 SOFTWARE

BASIC Faster & Better Library Disk	$2
BASIC Faster & Better Demo Disk	$2
BASIC I/O Demonstration Disk	2
TRSDOS 6.2 Utilities with Manual	9
Electric Pencil Word Processor	9

TRS-80 BOOKS

Games & Graphics for the TRS-80	$1
Inside Level II	1
Tandon 848-1 Service Manual	5

Add $2 shipping per order for books

AEROCOMP

2544 West Commerce St. Dallas, Texas 75212

SERVICE: 214-638-8886 INFORMATION 214-637-5400
TELEX: 882761
FAX: 214-634-8303

MasterCard · VISA · TeleCheck

"SERVING YOU SINCE 1980"

ORDER TOLL FREE!

M-F 9-7; Sat. 10-3

1-800-527-0347 AD Y1

Have your American Express, MasterCard or Visa ready. We will not charge your card until the day we ship your order. Mail orders are welcome. Money orders are accepted as well as your company and personal checks as long as they are bank printed and have your address and telephone number. We will ship surface COD with no deposit on most items, but all COD's require cash or a Cashier's Check on delivery. Texas residents add State Sales Tax. No tax collected on out of state shipments. There is a one year warranty (unless otherwise stated) on all hardware items against defects in materials or workmanship. Your satisfaction is guaranteed on hardware products. If you are not satisfied, for any reason, call us within 30 days of receipt and we will cheerfully refund your money (less shipping). All original materials must be intact and undamaged, as well as the original container. This offer does not apply to software. Defective software will be replaced. No other software warranty applies. Prices and specifications are subject to change without notice. Any returns must have our authorized RMA number on the label to be accepted.

A WORKAHOLIC'S DREAM COME TRUE

THE MODEL 100

by Sylvia Cary

Workaholism must be inherited, because ever since I was a little girl I always loved the idea of being able to do two things at once: Listen to the radio and do my homework; eat dinner and watch TV; talk on the phone and polish my nails. One year when I was still living in Manhattan, I read all of Dostoievski, Tolstoy, and Theodore Dreiser going back and forth to work on the Madison Avenue bus. Later, as a new young mother, I used to nurse my daughter and read Time magazine at the same time!

The idea of having a couple of hours to kill without doing something (preferable two somethings) constructive is so horrifying to me that I always make it a point to be prepared such an emergency. Since I'm a writer, I've turned my purse into a traveling office. In it I have pens, pencils, little notebooks for scribbling down "ideas," 3x5 filecards for research notes, on-going lists of "things to get" and "things to do," a folded-up draft of my current work-in-progress, a radio and, of course, a book to read "just in case" I get stuck in an elevator, or have a flat tire and have to wait for the auto club, or find myself imprisoned and have to wait for somebody to come and bail me out. At least I know I'll be able to utilize the time well.

Knowing this about me, imagine my thrill one afternoon in early 1984 when I hopped onto the Lexington Avenue subway in New York, sat down, gazed out into space (wondering how best to use the next twenty minutes of commuting time), and noticed the young man sitting across the aisle from me. He was doing something remarkable. He was typing away into what looked like a little black typewriter. It was the size of a student's spiral notebook, and only about two and a half inches high. Instantly, I fell in love -- not with the man, with the machine! I knew I had to have one, whatever it was. (Later I was to find out that it was the Radio Shack Model 100 computer with 24K, quite enough memory for a decent little story). Here was a man doing something after my own heart: Two things at once. Instead of just commuting, or at best reading graffiti, he was both commuting AND writing. With that wonderful little machine, he could actually write while in violent motion, something I'd tried to do for years on subways, buses, cars and trains, and had given up on. But here he was, doing it. I was smitten for sure!

Soon after that ride on the subway, I moved back to California, and I fell in love again -- this time with a human being, who happened to be a computer programmer and fellow workaholic. It was a match made in heaven! When he was romancing me, did we go dancing? Did we go frolicking on the beach? Did we take couples' workshops on how to communicate, share feelings and increase intimacy? No way! Our idea of a "hot" date was to go to a coffee shop, he with a computer magazine, and I with a book, and we'd sit there for hours and nibble our food and sip our coffee -- and READ. Two things at once!

When he proposed to me, did he buy me a diamond ring? No. Much better! He bought me my heart's desire, a Radio Shack Model 100 computer. It didn't come in a little black velvet box, it came in a sleek, black vinyl case, with velcro flaps for quick closing. What a dandy little machine. With only four double-A batteries, it can sing on for weeks. And it's light. I just put it in its case, slip it into a cloth student book bag, hook the straps over my shoulder, and I'm on my way. I take it everywhere -- coffee shops, phonebooths, meetings, classes, interviews. I can sit outside in the California sun and transcribe tape-recorded interviews into the Model 100, or type up first drafts of new projects, and wait until I feel like going inside to print it out.

And for a wedding anniversary, did my husband buy me a sexy nightgown? Indeed, no! He bought me an adorable, sexy little disk drive for the Model 100 instead. Now I can take even take it on long trips and not worry about running out of memory. On a train from Los Angeles to New York for a visit (I like trains), I sat in my bouncy little roomette and typed away with no worries for three-and-a-half days. My productivity, needless to say, has increased greatly. I've always considered myself a slow writer, but in the past year and a half, thanks to the help of my Model 100, I've written two books and the scripts for four educational movies.

A few months ago my husband and I had a "date" together at one of our favorite romantic spots, the local Mall. After a quick tour of a department store followed by a brief glance in the window of a jewelry store, we ended up at Radio Shack. It was there that I saw something that took my breath away: A newer, younger, sleeker, lighter, more powerful version of the Model 100, the Model 200! I fell in love all over again. "Honey," I cooed, snuggling up to my husband. "About the Model 200...Remember, I have a birthday coming up..."

Any other husband would have said, "But you already have one of those machines, what do you need a second one for?" But not *my* husband. He understood perfectly. He understood that for somebody like me who loves to do two things at once, to have one portable computer along with me at the coffee shop is good, but to have two portable computers along with me at the coffee shop is even better.

THE SWAP MEET

WANTED: Any literature, such as owner's or tech manuals for the Model III in Spanish. Basic manuals or anything of likely value to beginners. Will buy or pay to duplicate. Call Harold May. (312) 325-1910 collect.

WANTED: Any software/hardware to allow the Model III or IV to communicate with H.P. xy recorders, analog or digital type.
John Greenland. Box 171,
Kelligrews, New Foundland. A0A 2T0 Canada

FOR SALE: TRS-80 SOFTWARE, Models 1/3/4/4P/4D. Many useful programs. Economical prices.
Send $3 for listing.
Practical Programs, 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

FOR SALE: Model 4 with dual floppy, 128K, green monitor; keyboard needs work; will include software with original docs. Make a reasonable offer.
Paul Guglielmotti, 14512 Pacific Ave.
Baldwin Park, CA. 91706 (818) 962-8233

WANTED: Used Model 100 or 102 in good condition with at least 24K. Also need single/double sided disk drive for above. Must be priced reasonably.
Lance Wolstrup, 20311 Sherman Way, Suite 221.
Canoga Park, CA. 91306

FOR SALE: Printer Buffer, Centronics Port compatible (IBM PC + others), 64 KBytes (25 pages), Reset/Bypass/Copy buttons, 8 LED indicators (status + memory fullness), 5x7x2 inch metal case, 2 pounds, AC/DC with Power supply, builtin Selfcheck, 1 year guarantee, includes shipping - $119. Call/write:
Practical Programs, 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

WANTED: To complete collection, I am looking for SOFTSIDE Nov 80, Dec 80, and Jan 81. Also looking for any and all issues of PROG 80.
Lance Wolstrup. 20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

WANTED: LISP, PROLOG, APL, ADA, C+ +, etc. for Model III/4. Also RS Service Repair Manuals for Mod III/4, Forth Robot Arm for Mod III/4. Wish to contact folks who are in to Robotics, AI, Neural Networks, Nanotechnology.
R. Yves Breton. C.P. 95, Stn. Place D'Armes
Montreal, Quebec, Canada H2Y 3E9

WANTED: I am looking for back Issues of the Radio Shack Computer Catalog to complete my collection. I specifically need the following issues: RSC-1, 3, 13, 18E, 19E, 20E, 21E, and Software Buyers Guide 1st Edition. Any assistance would be greatly appreciated.
Roy Beck. 2153 Cedarhurst Dr.
Los Angeles, CA. 90027

FOR SALE: PD GOOD GAMES FOR MODEL I/III.
GAMEDISK#1: amazin/bas (maze), blazer/cmd (arcade), breakout/cmd (break down the walls), centipede/cmd (arcade), elect/bas (a simulation of the 1980 election), madhouse/bas (adventure), othello/cmd (board), poker/bas (almost better than going to las vegas - well, cheaper!!), solitr/bas (great solitaire card game). towers/cmd (puzzle game).
GAMEDISK#2: crams2/cmd (chase), falien/cmd (arcade), frankadv/bas (adventure), iceworld/bas (adventure game), minigolf/bas (putt-putt on the trs-80), pingpong/cmd (1 or 2 player arcade game), reactor/bas (simulation), solitr2/bas (another good solitaire card game), stars/cmd (2 player race game), trak/cmd (maze game).
GAMEDISK#3: ashka/cmd (d&d), asteroid/cmd (arcade), crazy8/bas (card game), french/cmd (space invaders In french), hexapawn (board game), hobbit/bas (adventure game), memalpha (adventure game), pyramid/bas (good solitaire card game), rescue/bas (arcade game), swarm/cmd (arcade game).

Price per disk: $5.00 (U.S.)
or get all 3 disks for $12.00 (U.S.)
TRSTimes - PD GAMES. 20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

CLOSE #2

In closing this issue, our 14th, we would like to thank each one of the wonderful people who helped make it possible.

David Berg's STANTEST/BAS should be a big help to all our overworked and underpaid school teachers with a TRS-80, and there are many out there.

Dr. Alien Jacobs, in his usual in-depth fashion, reviews version 2 of GIF4MOD4.

The HINTS & TIPS section feature goodies from Carol Welcomb, Robert Doerr, Don Bergthold, David Welsh, and Ye Faithful Editor. I thoroughly enjoy these, as they are short and, for the most part, easy to comprehend. If you have something to share, but do not feel up to the task of writing a complete article, you might keep this format in mind.

The ASSEMBLY 101 column continues with more Z-80 Assembly language programming. This time, because of sheer length, it was necessary to split the installment in half. The program started in this issue, TRSLABL4/SRC, is one that I use at TRSTimes on a daily basis. It is a handy program when you need to address a few individual letters. The rest of the program code will be presented in the May/Jun issue.

Amazing Roy Beck presents part 2 of his adventures of installing several operating systems, including CP/M, on his hard disk. Again, because of space limitations, it was necessary to defer the conclusion (part 3) until the next issue.

Sylvia Cary presents a humorous view of what happens when two computer-crazed workaholics get together. Does this sound familiar?

Next issue, besides concluding the above mentioned articles, will feature a fine installment by Jeff Joseph about using the Z-80 alternate registers. For the game/puzzle aficionados we will bring a basic program for Model 4 based on the moves on the chess queen. Watch out for this one - the TRS-80 plays a deadly game. Also planned is a look at LS-DOS 6.3.1, as well as other goodies.

Finally, to all the readers, please send us your articles, programs, hints & tips, reviews and letters for possible publication. We do need your participation.

See you in May.....

Lance W.

Our new LS-DOS 6.3.1 release has a little something for everyone

- ☆ The DATE command, "Date?" prompt on boot, and the @DATE SVC now support a date range of 32 years; from **January 1, 1980 through December 31, 2011.**
- ☆ **Enable or disable the printer time-out** and error generation with SYSTEM (PRTIME=ON|OFF)
- ☆ Customize the display of the time field in the DIR command to display **12-hr or 24-hr clock time** with SYSTEM (AMPM=ON|OFF).
- ☆ Both ASCII and hexadecimal display output from the LIST command is **paged a screen at a time.** Or run it non-stop under your control.
- ☆ MEMORY displays (or prints) the status of switchable memory banks known to the DOS, as well as a **map of modules** resident in I/O driver system memory and high memory.
- ☆ Specify SYSTEM (DRIVE=d1,SWAP=d2) to **switch drive d1 for d2.** Either may be the system drive, and a Job Control Language file may be active on either of the swapped drives.
- ☆ The TED text editor now has commands to **print the entire text buffer,** or the contents of the first block encountered.
- ☆ Have extended memory? The SPOOL command now permits the BANK parameter entry to range from 0-30 instead of 0-7.
- ☆ **Alter the logical record length** of a file with "RESET filespec (LRL=n)"
- ☆ Specify "RESET filespec (DATE=OFF)" to restore a file's directory entry to the old-style dating of pre-6.3 release. Specify "RESET filespec (DATE=ON)" to establish a file's directory date as that of the **current system date and time.**
- ☆ Felt uncomfortable with the *alleged* protection scheme of 6.3? **LS-DOS 6.3.1 has no anti-piracy protection!** MISOSYS trusts its customers to honor our copyrights.
- ☆ Best of all, **an LS-DOS 6.3.1 diskette is available as a replacement disk for $15** (plus $2 S&H in US). There's no need to return your current master.
- ☆ The 6.3.1 diskette comes with a 30-day warranty; written customer support is available for 30 days from the purchase date. Versions for the Model 4 and Model II/12 are available. **If you do not already have an LS-DOS 6.3.0, order the 6.3.1 Upgrade Kit with 90 days of customer support for $39.95 (+$2 S&H).**

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 22170-0239
703-450-4181 [orders to 800-MISOSYS (647-6797)]

MISOSYS SPECIALS OF THE MONTH

PRO-WAM Mister ED Application Pack half-off until March 31st

Mister ED is loaded with editor applications. All are full screen which make your editing jobs easy. Best of all, these are PRO-WAM applications so they can pop up even when you are using other Model 4 programs.

Mister ED includes: DED to edit disk sectors; FED, to edit file records; MED, to edit memory (even banked); VED, to edit the video screen; and TED, similar to TED/CMD to edit text files.

Only $19.98 + $3 S&H until March 31st.

SUPER UTILITY PLUS - The greatest floppy disk utility ever written for the TRS-80, now even greater at $10 off - Only $24.95 + $4 S&H!

SU+ is completely menu-driven and is configurable for all the popular TRS-80 operating systems. SU+ removes or decodes passwords, reformats a disk without erasing the data, fixes problems, backs up most protected disks, etc. SU+ has over 65 functions and features. Too many to describe! Does not work on hard disks. SU+ does not support Newdos/80 double-sided disks.

Specify Model I/III or 4. $24.95 + $4 S&H until March 31st.

ATTENTION TRSDOS 1.3. USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! MORE PUNCH!!

While maintaining 100% compatibility to TRSDOS 1.3., this DOS upgrade advances TRSDOS 1.3. into the 90's!
SYSTEM 1.5. supports 16k-32k bank data storage and 4MGHZ clock speed (4/4P/4D).

DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG = Y/N	CREATES CONFIG BOOT UP FILE	DATE = Y/N	DATE BOOT UP PROMPT ON or OFF
TIME = Y/N	TIME BOOT UP PROMPT ON or OFF	CURSOR = 'XX'	DEFINE BOOT UP CURSOR CHAR
BLINK = Y/N	SET CURSOR BOOT UP DEFAULT	CAPS = Y/N	SET KEY CAPS BOOT UP DEFAULT
LINE = 'XX'	SET *PR LINES BOOT UP DEFAULT	WP = d.Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
ALIVE = Y/N	GRAPHIC MONITOR ON or OFF	TRACE = Y/N	TURN SP MONITOR ON or OFF
TRON = Y/N	ADD an IMPROVED TRON	MEMORY = Y/N	BASIC FREE MEMORY DISPLAY MONITOR
TYPE = B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF	FAST	4 MGHZ SPEED (MODEL 4'S)
SLOW	2 MGHZ SPEED (MODEL III'S)	BASIC2	ENTER ROM BASIC (NON-DISK)
CPY (parm,parm)	COPY/LIST/CAT LDOS TYPE DISKS	SYSRES = H/B/'XX'	MOVE/SYS OVERLAY(s) TO HI/BANK MEM
SYSRES = Y/N	DISABLE/ENABLE SYSRES OPTION	MACRO	DEFINE ANY KEY TO MACRO
SPOOL = H/B.SIZE	SPOOL is HIGH or BANK MEMORY	SPOOL = D.SIZE = 'XX'	LINK MEM SPOOLING TO DISK FILE
SPOOL = N	TEMPORARILY DISABLE SPOOLER	SPOOL = Y	REACTIVATE DISABLED SPOOLER
SPOOL = RESET	RESET (NIL) SPOOL BUFFER	SPOOL = OPEN	OPENS, REACTIVATES DISK SPOOLING
SPOOL = CLOSE	CLOSES SPOOL DISK FILE	FILTER *PR.ADLF = Y/N	ADD LINE FEEDS BEFORE PRINTING 0DH
FILTER *PR.IGLF	IGNORES 'EXTRA' LINE FEEDS	FILTER *PR.HARD = Y/N	SEND 0CH to PRINTER (FASTEST TOF)
FILTER *PR.FILTER	ADDS 256 BYTE PRINTER FILTER	FILTER *PR.ORIG	TRANSLATE PRINTER BYTE TO CHNG
FILTER *PR.FIND	TRANSLATE PRINTER BYTE TO CHNG	FILTER *PR.RESET	RESET PRINTER FILTER TABLE
FILTER *PR.LINES	DEFINE NUMBER LINES PER PAGE	FILTER *PR.WIDTH	DEFINE PRINTER LINE WIDTH
FILTER *PR.TMARG	ADDS TOP MARGIN to PRINTOUTS	FILTER *PR.BMARG	ADDS BOTTOM MARGIN to PRINTOUT
FILTER *PR.PAGE	NUMBER PAGES, SET PAGE NUMBER	FILTER *PR.ROUTE	SETS PRINTER ROUTING ON or OFF
FILTER *PR.TOF	MOVES PAPER TO TOP OF FORM	FILTER *PR.NEWPG	SET DCB LINE COUNT TO 1
FILTER *KI.ECHO	ECHO KEYS to the PRINTER	FILTER *KI.MACRO	TURN MACRO KEYS ON or OFF
ATTRIB:d.PASSWORD	CHANGE MASTER PASSWORD	DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using the new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as a new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key,even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files, ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files and disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg: LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIBDVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

By special arrangement with GRL Software,
SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.

ORDER YOUR COPY TODAY!

Send $39.95 (U.S. funds) to:

TRSTimes - SYSTEM 1.5.
20311 Sherman Way, Suite 221
Canoga Park, CA. 91306